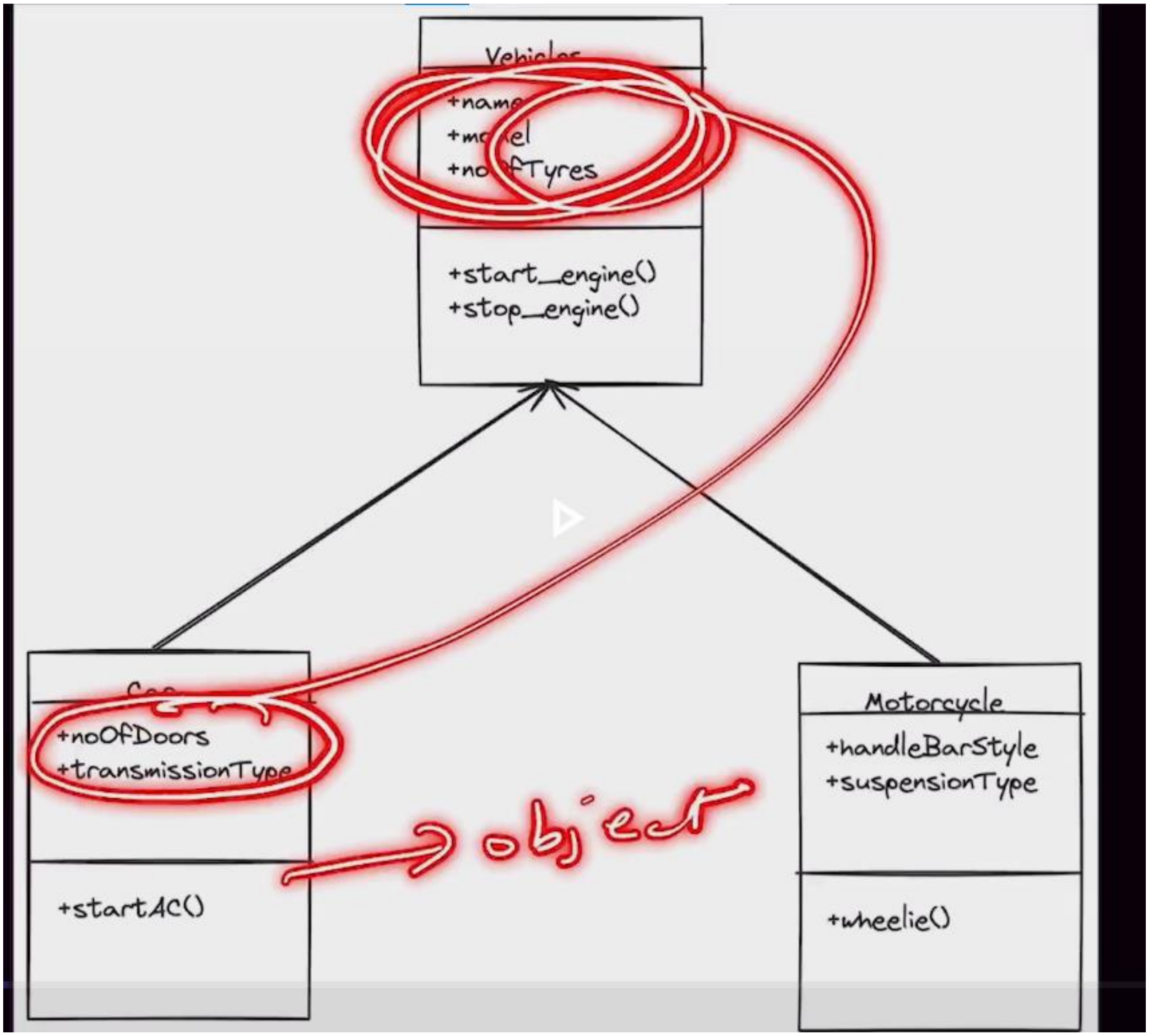
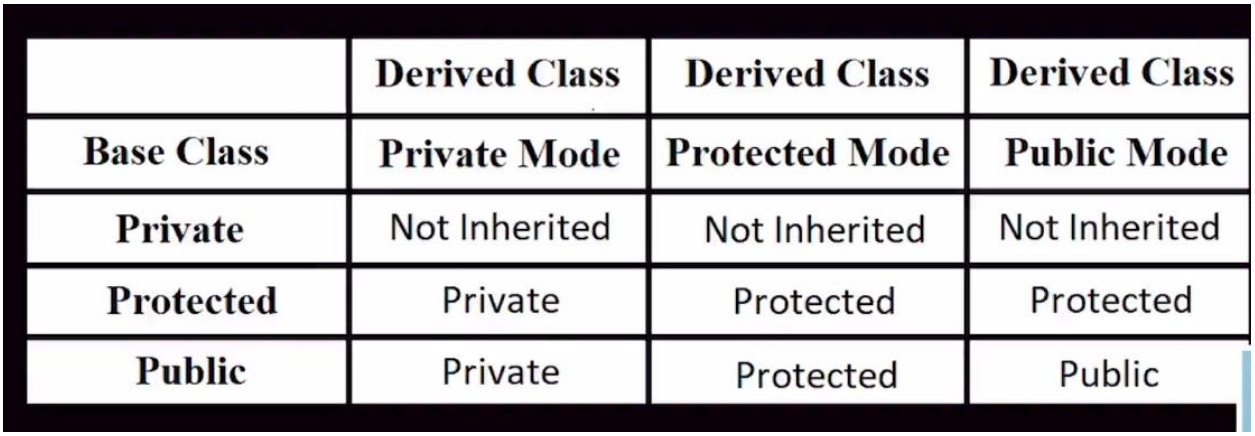
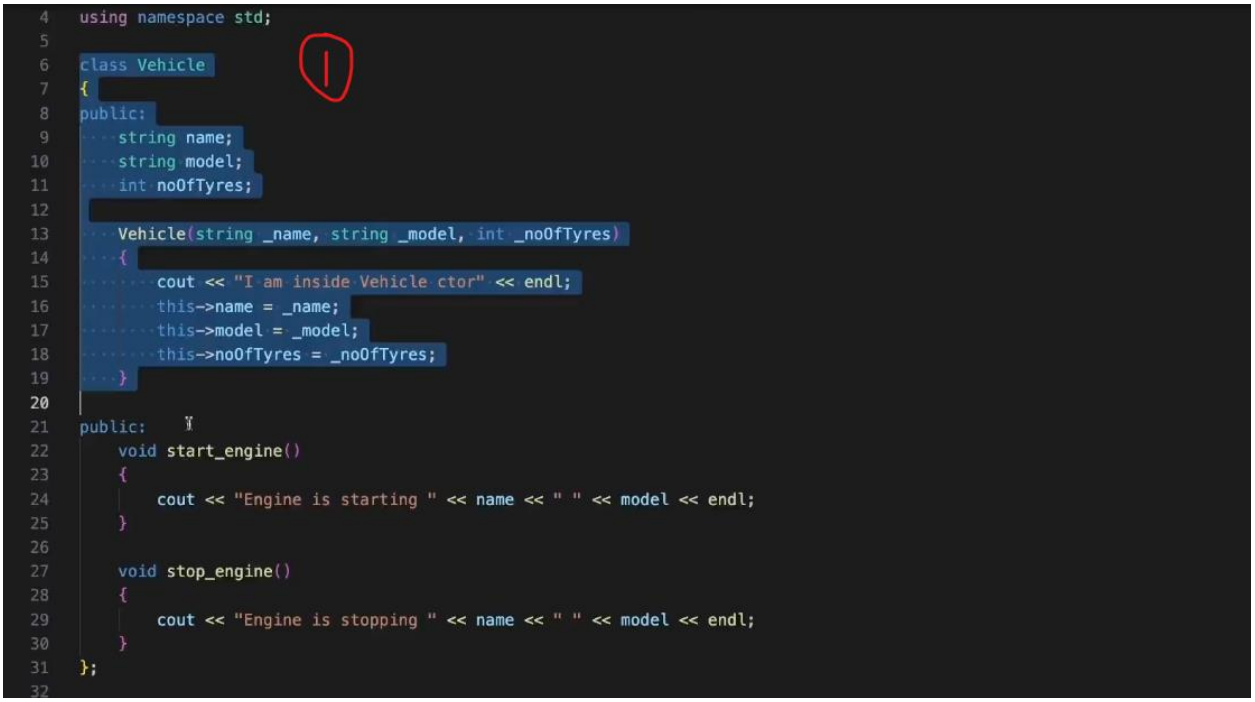
6. Implementing Inheritance in C++

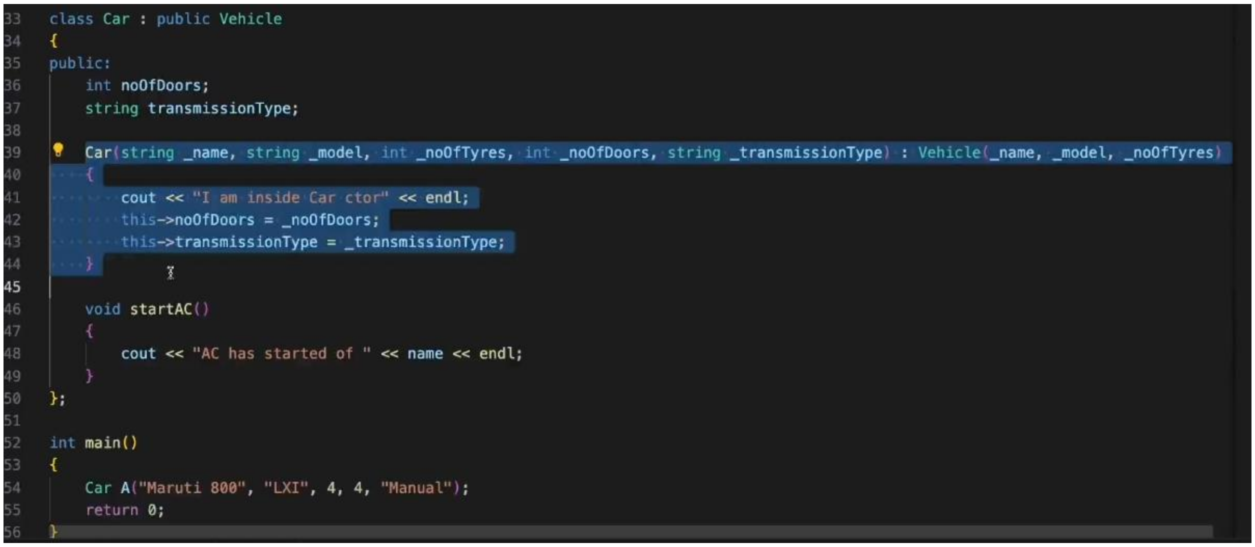




	Derived Class	Derived Class	Derived Class
Base Class	**Private Mode**	**Protected Mode**	**Public Mode**
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Private	Protected	Protected
Public	Private	Protected	Public

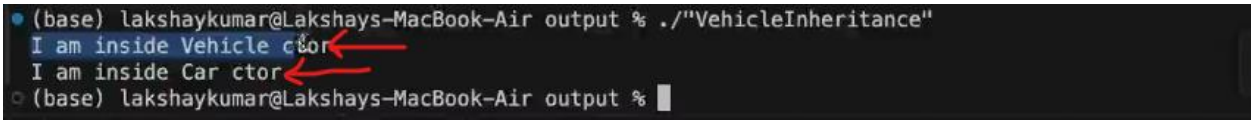


```cpp
using namespace std;

class Vehicle
{
public:
    string name;
    string model;
    int noOfTyres;

    Vehicle(string _name, string _model, int _noOfTyres)
    {
        cout << "I am inside Vehicle ctor" << endl;
        this->name = _name;
        this->model = _model;
        this->noOfTyres = _noOfTyres;
    }

public:
    void start_engine()
    {
        cout << "Engine is starting " << name << " " << model << endl;
    }

    void stop_engine()
    {
        cout << "Engine is stopping " << name << " " << model << endl;
    }
};

```



```cpp
class Car : public Vehicle
{
public:
    int noOfDoors;
    string transmissionType;

    Car(string _name, string _model, int _noOfTyres, int _noOfDoors, string _transmissionType) : Vehicle(_name, _model, _noOfTyres)
    {
        cout << "I am inside Car ctor" << endl;
        this->noOfDoors = _noOfDoors;
        this->transmissionType = _transmissionType;
    }

    void startAC()
    {
        cout << "AC has started of " << name << endl;
    }
};

int main()
{
    Car A("Maruti 800", "LXI", 4, 4, "Manual");
    return 0;
}
```

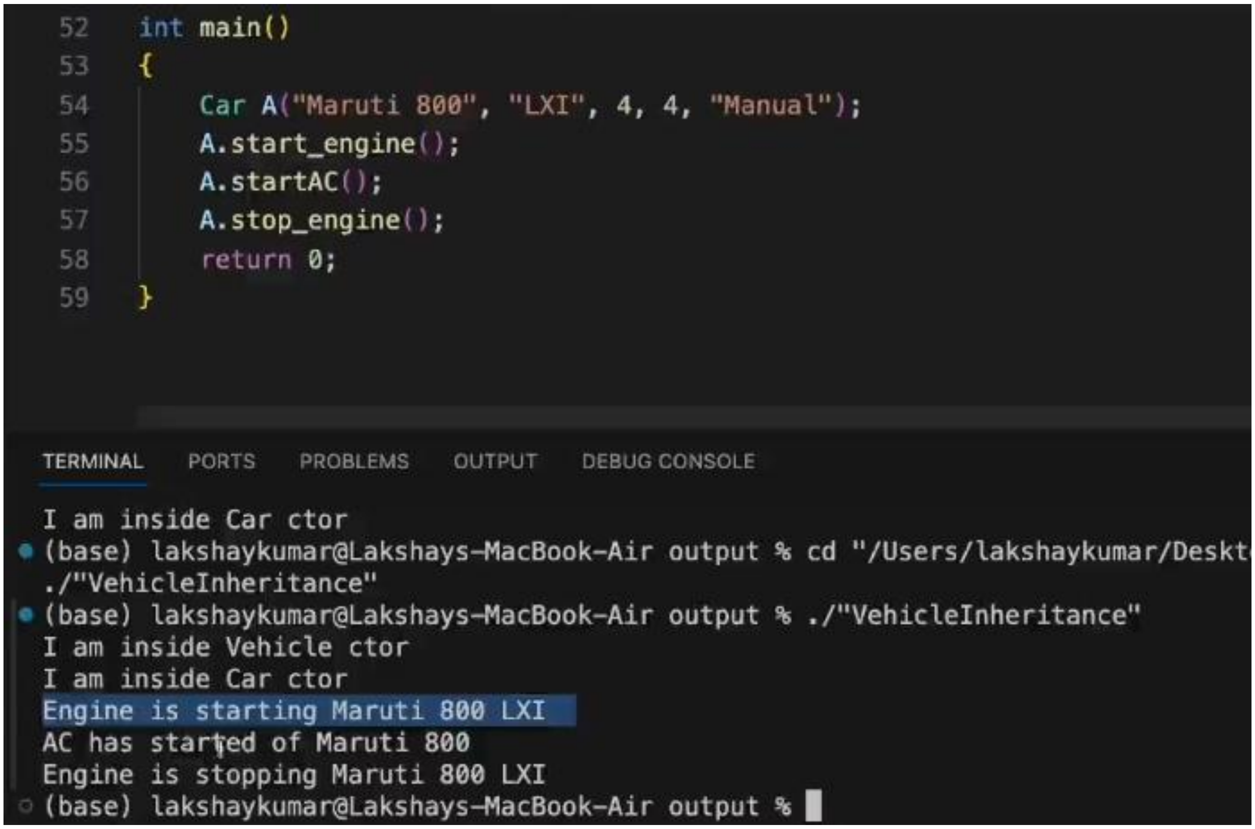
Output:



```
(base) lakshaykumar@Lakshays-MacBook-Air output % ./"VehicleInheritance"
I am inside Vehicle ctor
I am inside Car ctor
(base) lakshaykumar@Lakshays-MacBook-Air output %
```

Child class banane se pahle base class ka constructor call hota hai

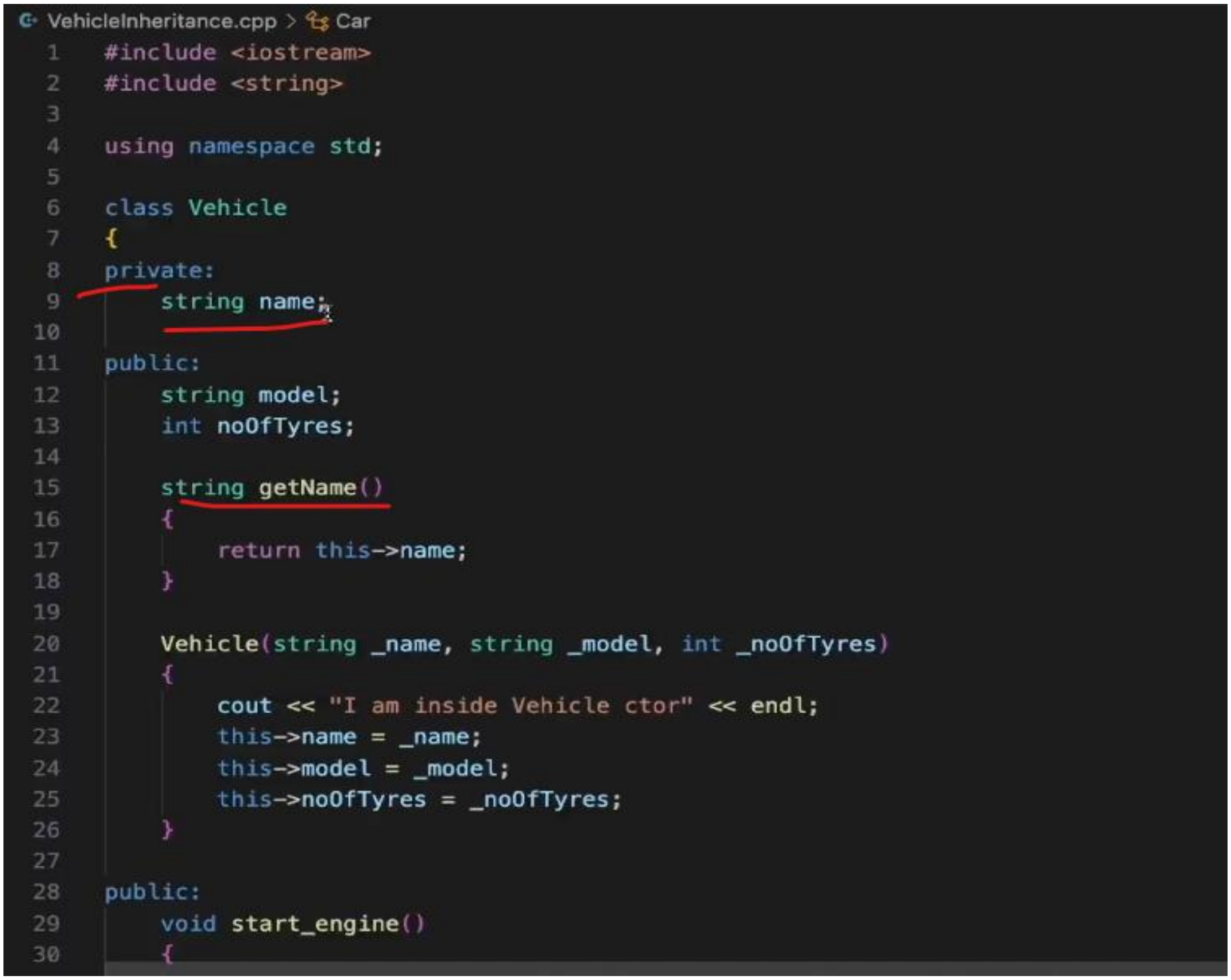
----



```cpp
int main()
{
    Car A("Maruti 800", "LXI", 4, 4, "Manual");
    A.start_engine();
    A.startAC();
    A.stop_engine();
    return 0;
}
```

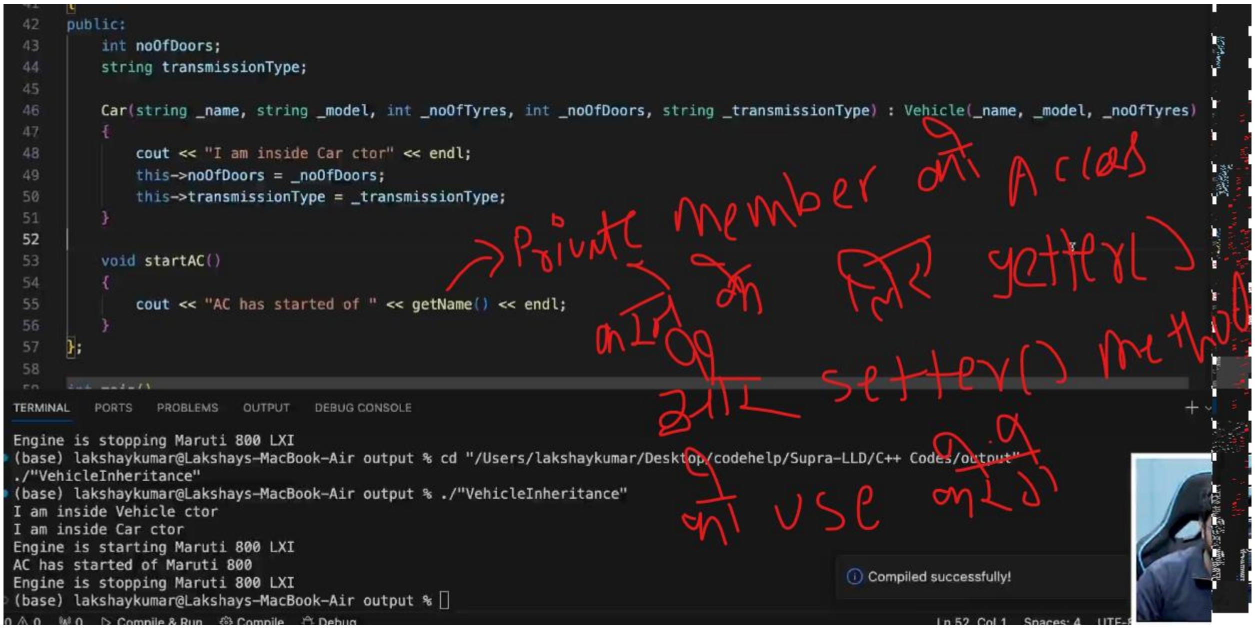
```
I am inside Car ctor
(base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Deskt
./"VehicleInheritance"
(base) lakshaykumar@Lakshays-MacBook-Air output % ./"VehicleInheritance"
I am inside Vehicle ctor
I am inside Car ctor
Engine is starting Maruti 800 LXI
AC has started of Maruti 800
Engine is stopping Maruti 800 LXI
(base) lakshaykumar@Lakshays-MacBook-Air output %
```

Base class ke private member ko inheriate kar rahe public child class mai toh ,member ko access nahi kar sakte child class ,agar private member ko access karna child mai toh GETTER () AUR SETTER () METHOD Ko use Karenge



VehicleInheritance.cpp > Car

```cpp
#include <iostream>
#include <string>

using namespace std;

class Vehicle
{
private:
    string name;

public:
    string model;
    int noOfTyres;

    string getName()
    {
        return this->name;
    }

    Vehicle(string _name, string _model, int _noOfTyres)
    {
        cout << "I am inside Vehicle ctor" << endl;
        this->name = _name;
        this->model = _model;
        this->noOfTyres = _noOfTyres;
    }

public:
    void start_engine()
    {
```



```cpp
public:
    int noOfDoors;
    string transmissionType;

    Car(string _name, string _model, int _noOfTyres, int _noOfDoors, string _transmissionType) : Vehicle(_name, _model, _noOfTyres)
    {
        cout << "I am inside Car ctor" << endl;
        this->noOfDoors = _noOfDoors;
        this->transmissionType = _transmissionType;
    }

    void startAC()
    {
        cout << "AC has started of " << getName() << endl;
    }
};
```

→ Private member को A class
के बाहर getter()
और setter() method
द्वारा
की use करते

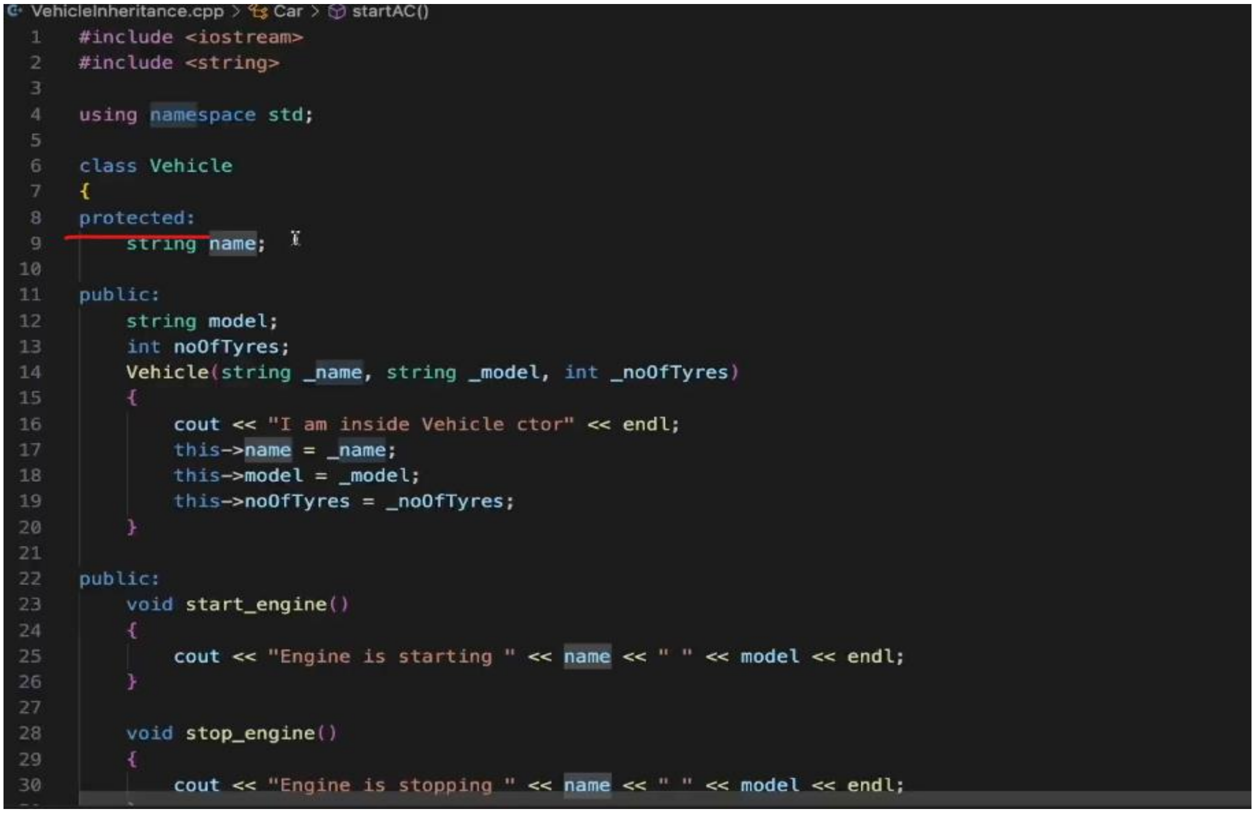
TERMINAL PORTS PROBLEMS OUTPUT DEBUG CONSOLE

```
Engine is stopping Maruti 800 LXI
(base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop/codehelp/Supra-LLD/C++ Codes/output"
./"VehicleInheritance"
(base) lakshaykumar@Lakshays-MacBook-Air output % ./"VehicleInheritance"
I am inside Vehicle ctor
I am inside Car ctor
Engine is starting Maruti 800 LXI
AC has started of Maruti 800
Engine is stopping Maruti 800 LXI
(base) lakshaykumar@Lakshays-MacBook-Air output % 
```

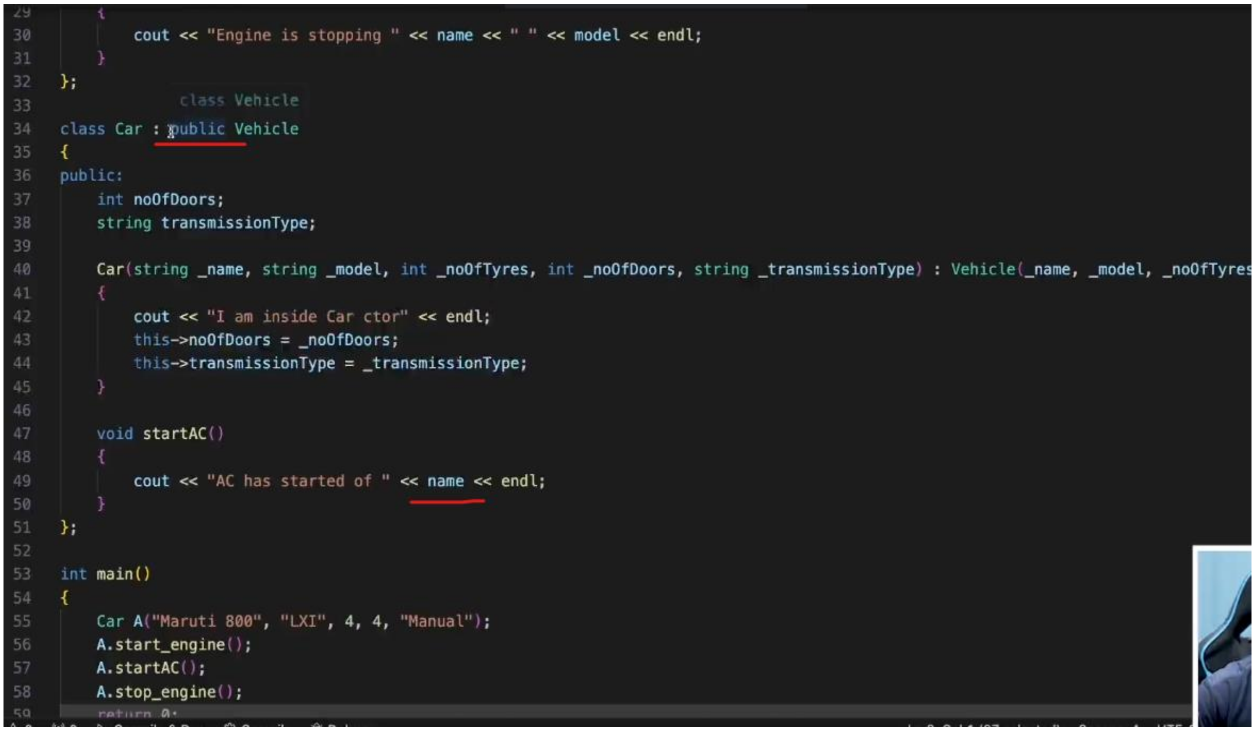
ⓘ Compiled successfully!

Ln 52, Col 1 Spaces: 4 UTF-

Base class ke Protected member ko child public mode class mai access karenge toh ,child class mai as protected member aa jayeag.

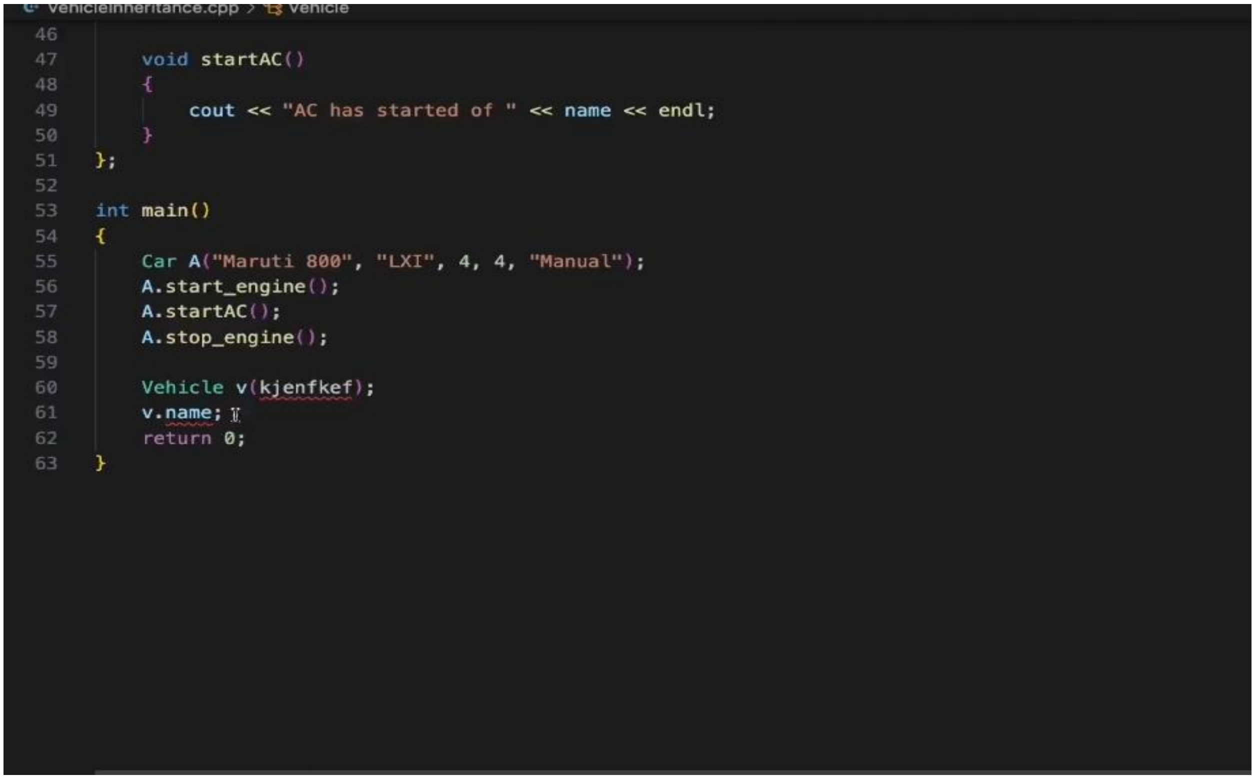


```cpp
#include <iostream>
#include <string>

using namespace std;

class Vehicle
{
protected:
    string name;

public:
    string model;
    int noOfTyres;
    Vehicle(string _name, string _model, int _noOfTyres)
    {
        cout << "I am inside Vehicle ctor" << endl;
        this->name = _name;
        this->model = _model;
        this->noOfTyres = _noOfTyres;
    }

public:
    void start_engine()
    {
        cout << "Engine is starting " << name << " " << model << endl;
    }

    void stop_engine()
    {
        cout << "Engine is stopping " << name << " " << model << endl;
```

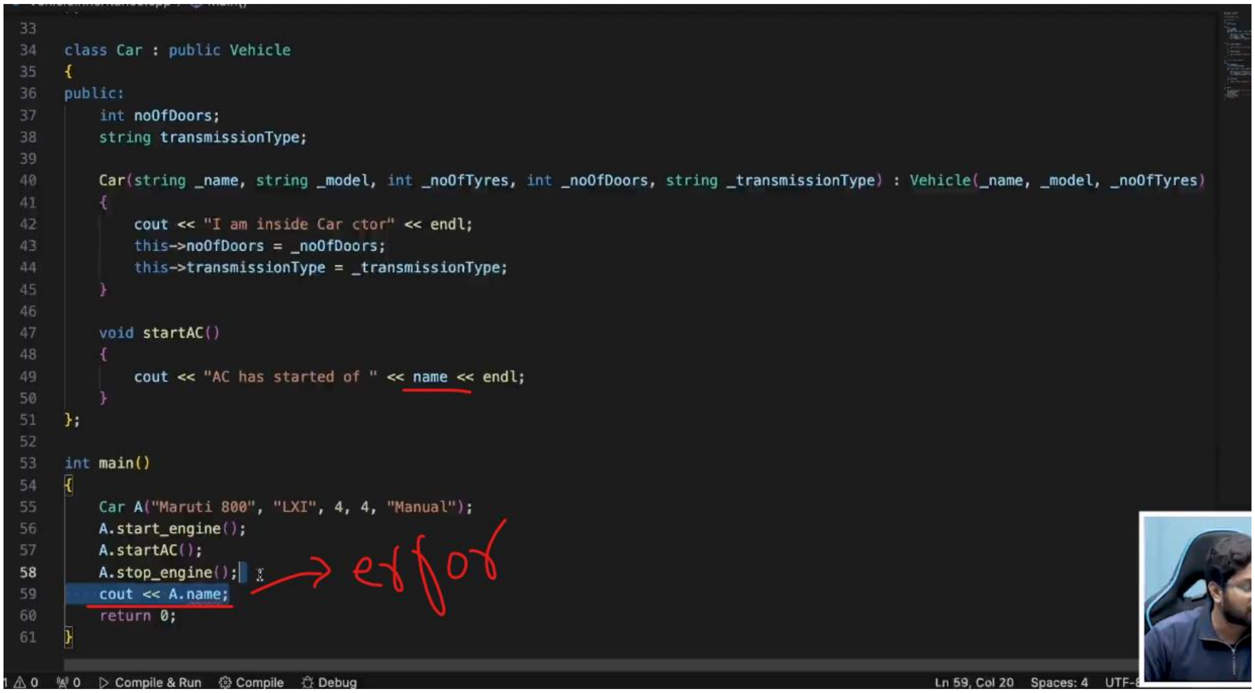


```cpp
    {
        cout << "Engine is stopping " << name << " " << model << endl;
    }
};

class Car : public Vehicle
{
public:
    int noOfDoors;
    string transmissionType;

    Car(string _name, string _model, int _noOfTyres, int _noOfDoors, string _transmissionType) : Vehicle(_name, _model, _noOfTyres
    {
        cout << "I am inside Car ctor" << endl;
        this->noOfDoors = _noOfDoors;
        this->transmissionType = _transmissionType;
    }

    void startAC()
    {
        cout << "AC has started of " << name << endl;
    }
};

int main()
{
    Car A("Maruti 800", "LXI", 4, 4, "Manual");
    A.start_engine();
    A.startAC();
    A.stop_engine();
    return 0;
```

Parent class se protected member child class mai inheriate toh ho jayega as protected member lekin private ke jais behave karega. Child class mai acces hoga lekin jab child ka object bana kar ya base class ka object bana ke karenge toh error aaega



```cpp

    void startAC()
    {
        cout << "AC has started of " << name << endl;
    }
};

int main()
{
    Car A("Maruti 800", "LXI", 4, 4, "Manual");
    A.start_engine();
    A.startAC();
    A.stop_engine();

    Vehicle v(kjenfkef);
    v.name;
    return 0;
}
```



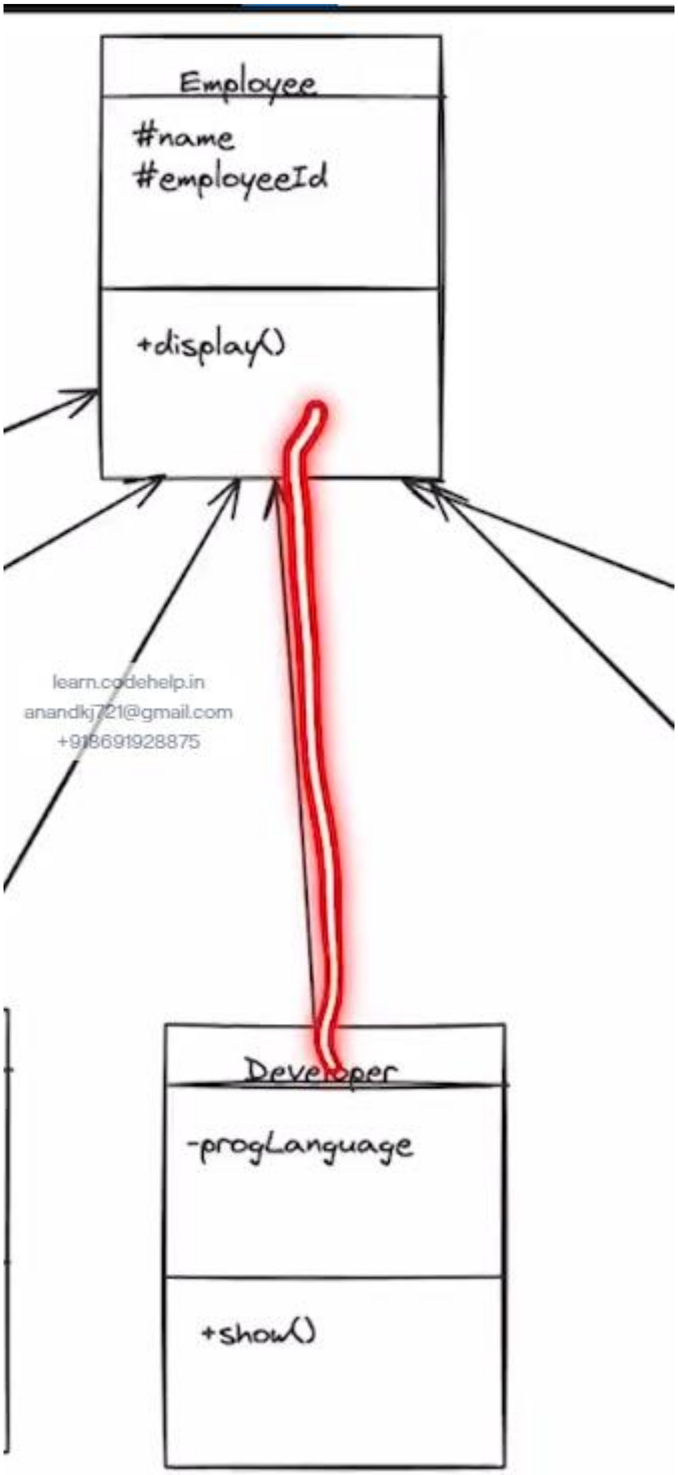
```cpp

class Car : public Vehicle
{
public:
    int noOfDoors;
    string transmissionType;

    Car(string _name, string _model, int _noOfTyres, int _noOfDoors, string _transmissionType) : Vehicle(_name, _model, _noOfTyres)
    {
        cout << "I am inside Car ctor" << endl;
        this->noOfDoors = _noOfDoors;
        this->transmissionType = _transmissionType;
    }

    void startAC()
    {
        cout << "AC has started of " << name << endl;
    }
};

int main()
{
    Car A("Maruti 800", "LXI", 4, 4, "Manual");
    A.start_engine();
    A.startAC();
    A.stop_engine();
    cout << A.name;
    return 0;
}
```

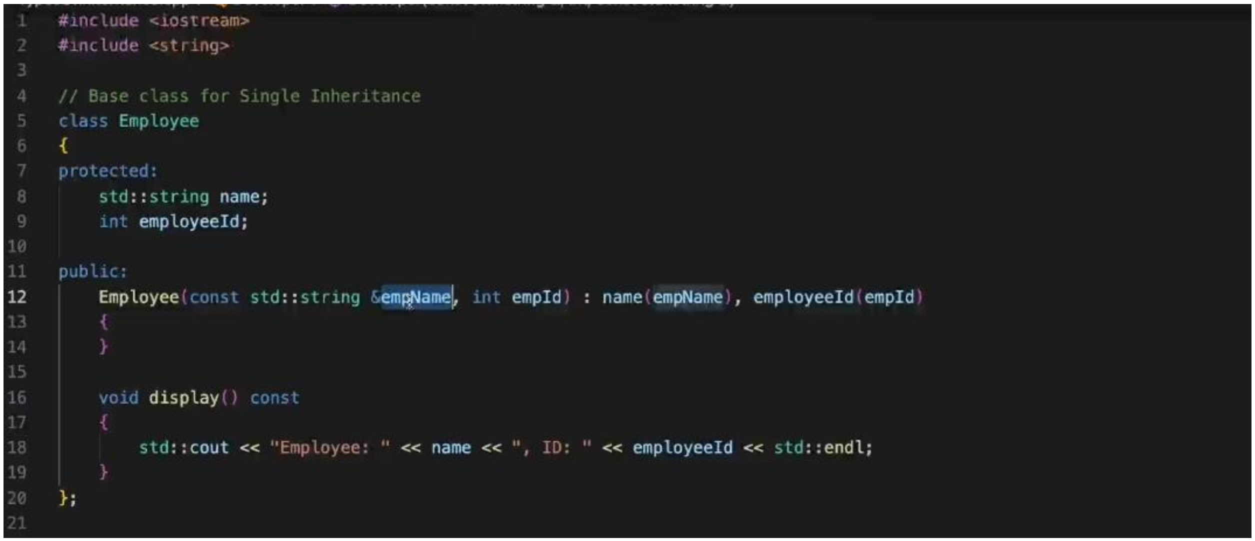
Protected member parent se child derived class mai aa jayega lekin main class access nahi hoga.

Base class ka constructor derived class ke constructor se pahle call hoga .lekin jab hum destructor call opposite hota hai,pahle derived class ka destructor call hoga uske badd mai Base class ka destructor

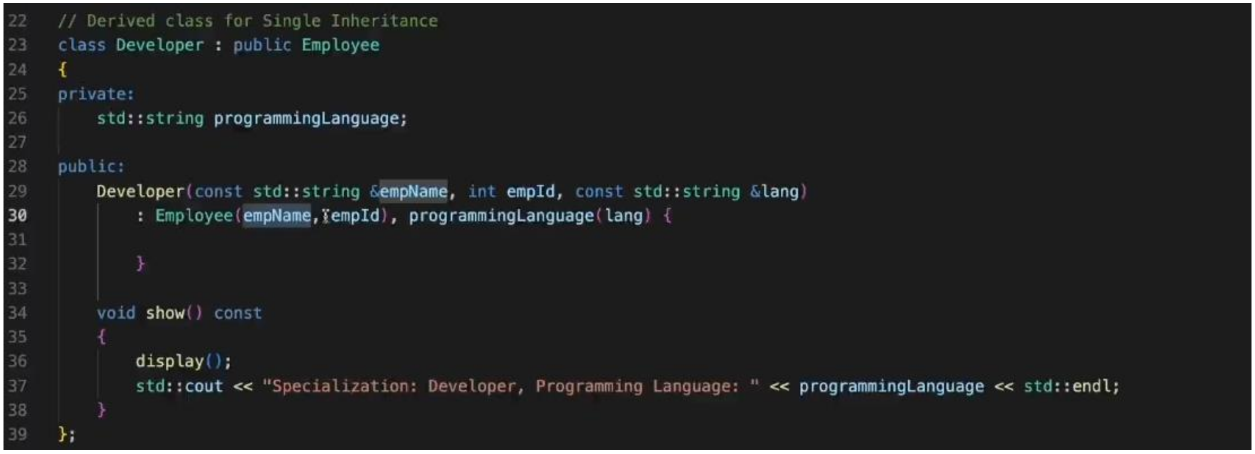
---

Single Inheritance

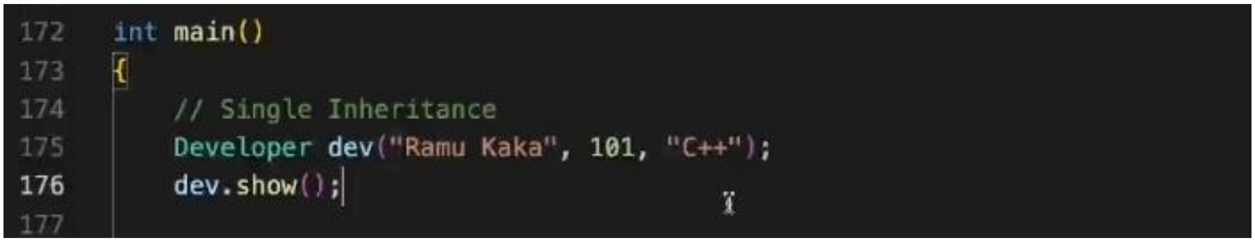




```cpp
#include <iostream>
#include <string>

// Base class for Single Inheritance
class Employee
{
protected:
    std::string name;
    int employeeId;

public:
    Employee(const std::string &empName, int empId) : name(empName), employeeId(empId)
    {
    }

    void display() const
    {
        std::cout << "Employee: " << name << ", ID: " << employeeId << std::endl;
    }
};

```

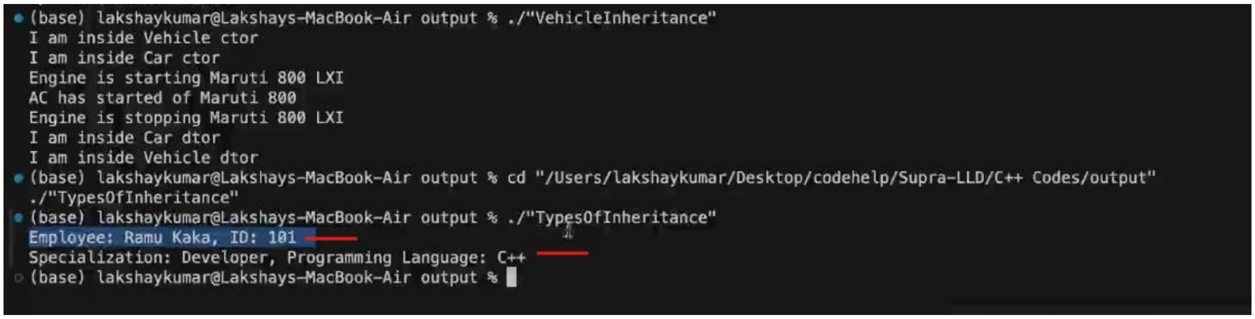


```cpp
// Derived class for Single Inheritance
class Developer : public Employee
{
private:
    std::string programmingLanguage;

public:
    Developer(const std::string &empName, int empId, const std::string &lang)
        : Employee(empName, empId), programmingLanguage(lang) {

        }

    void show() const
    {
        display();
        std::cout << "Specialization: Developer, Programming Language: " << programmingLanguage << std::endl;
    }
};
```



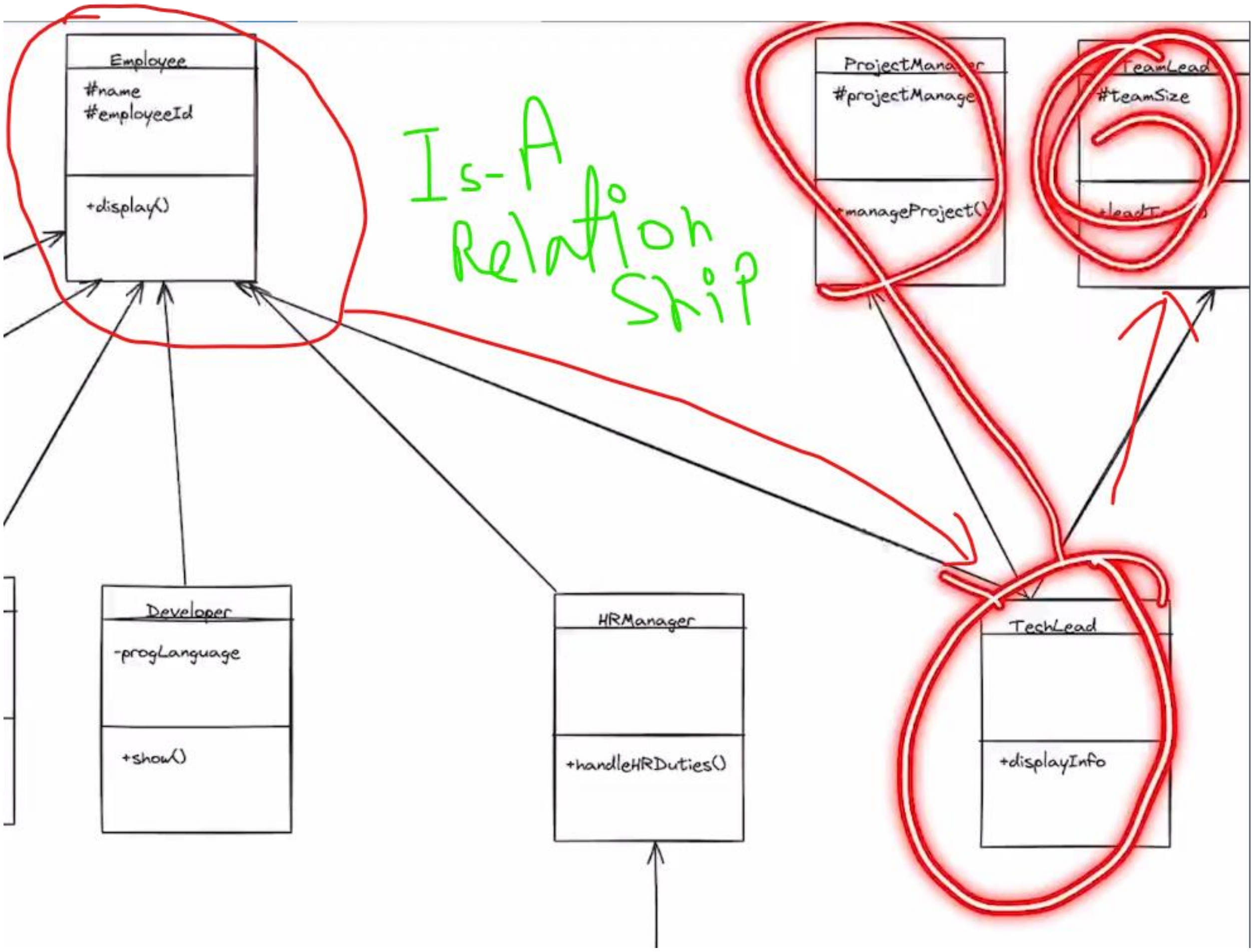
```cpp
int main()
{
    // Single Inheritance
    Developer dev("Ramu Kaka", 101, "C++");
    dev.show();

```

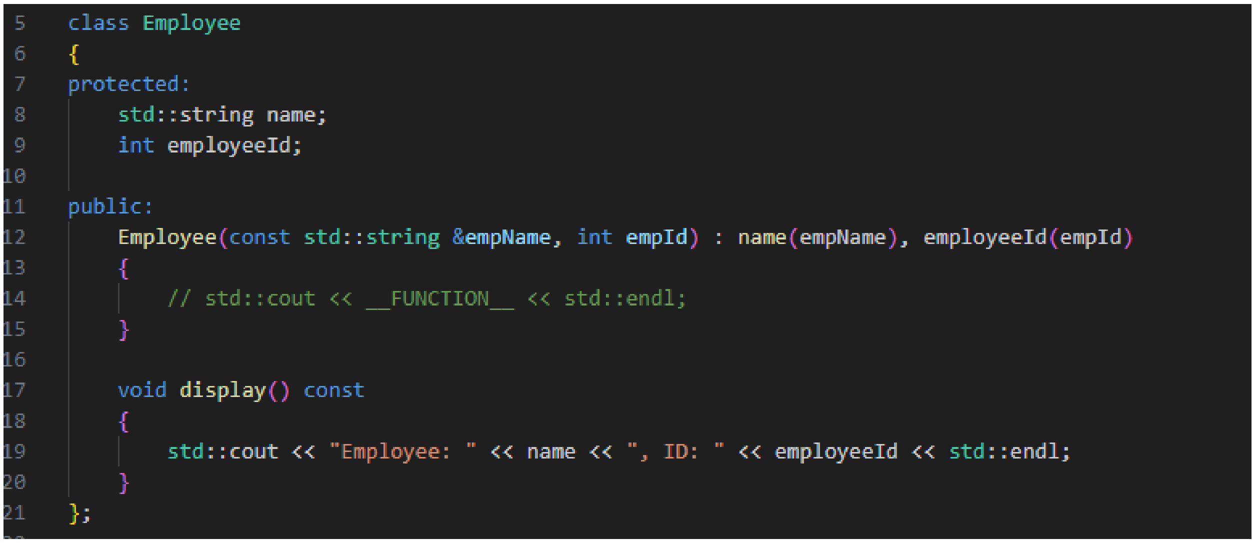
Output



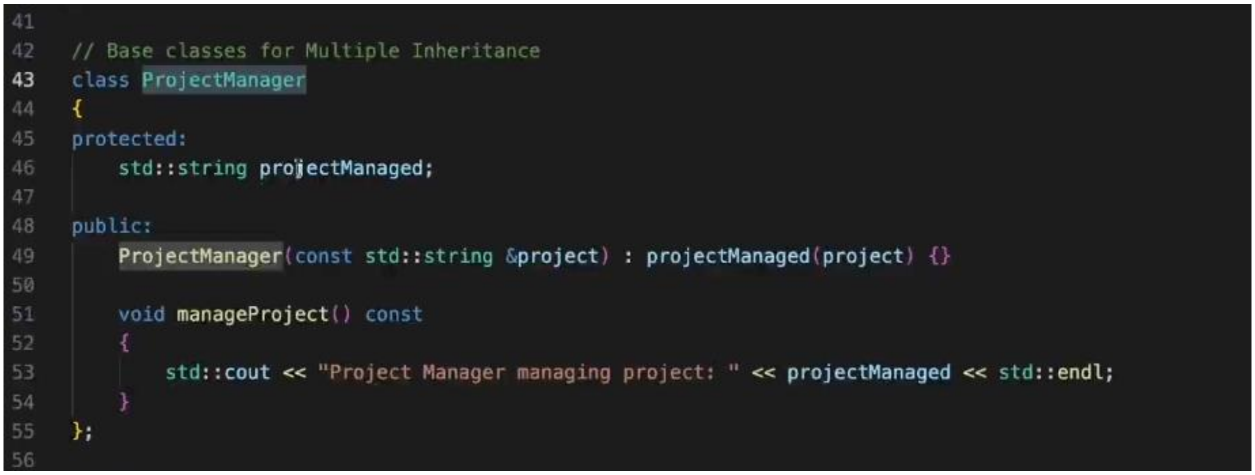
```
(base) lakshaykumar@Lakshays-MacBook-Air output % ./"VehicleInheritance"
I am inside Vehicle ctor
I am inside Car ctor
Engine is starting Maruti 800 LXI
AC has started of Maruti 800
Engine is stopping Maruti 800 LXI
I am inside Car dtor
I am inside Vehicle dtor
(base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop/codehelp/Supra-LLD/C++ Codes/output"
./"TypesOfInheritance"
(base) lakshaykumar@Lakshays-MacBook-Air output % ./"TypesOfInheritance"
Employee: Ramu Kaka, ID: 101
Specialization: Developer, Programming Language: C++
(base) lakshaykumar@Lakshays-MacBook-Air output %
```

Multiple Inheritance

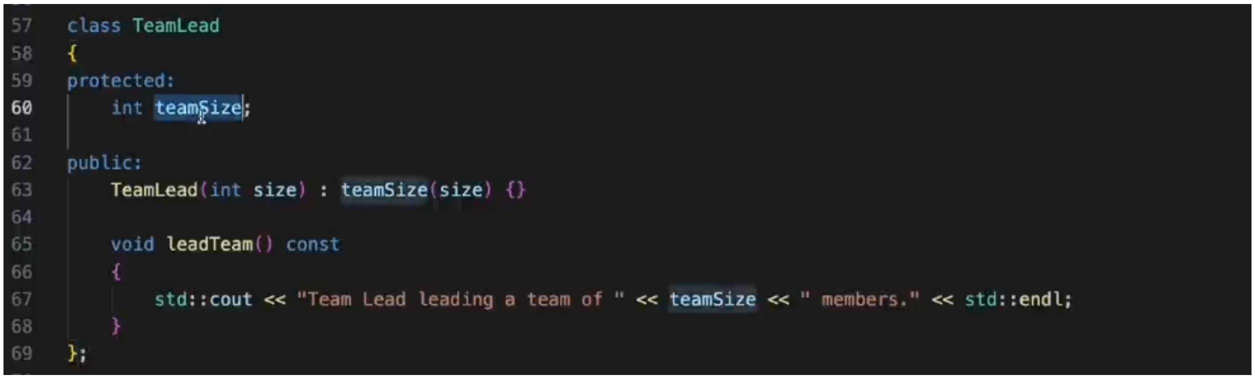




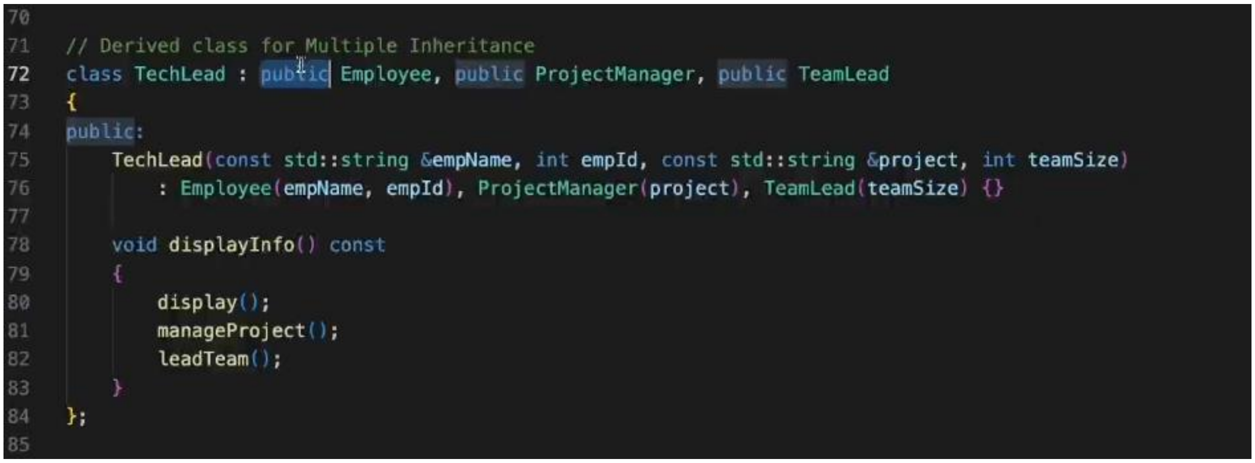
```
class Employee
{
protected:
    std::string name;
    int employeeId;

public:
    Employee(const std::string &empName, int empId) : name(empName), employeeId(empId)
    {
        // std::cout << __FUNCTION__ << std::endl;
    }

    void display() const
    {
        std::cout << "Employee: " << name << ", ID: " << employeeId << std::endl;
    }
};
```



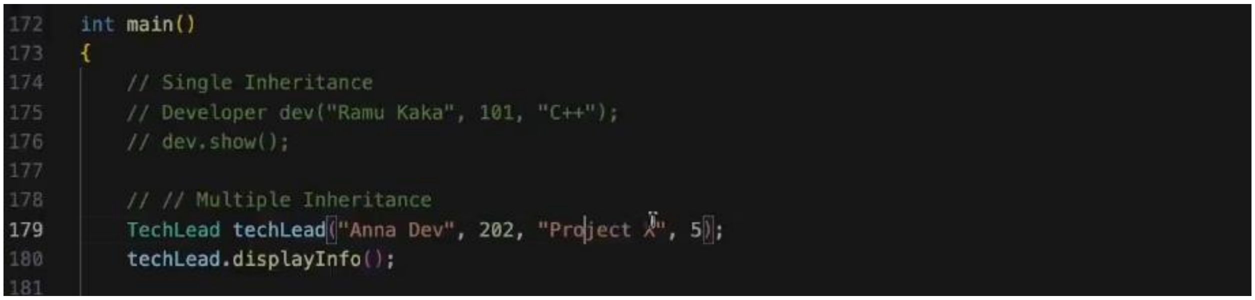
```
// Base classes for Multiple Inheritance
class ProjectManager
{
protected:
    std::string projectManaged;

public:
    ProjectManager(const std::string &project) : projectManaged(project) {}

    void manageProject() const
    {
        std::cout << "Project Manager managing project: " << projectManaged << std::endl;
    }
};

```



```
class TeamLead
{
protected:
    int teamSize;

public:
    TeamLead(int size) : teamSize(size) {}

    void leadTeam() const
    {
        std::cout << "Team Lead leading a team of " << teamSize << " members." << std::endl;
    }
};
```

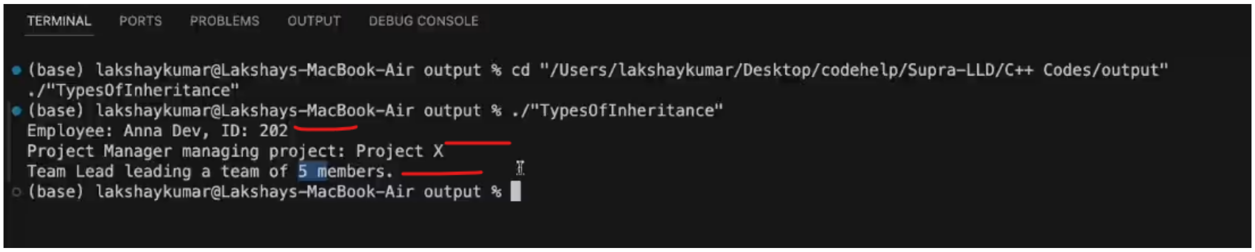


```

// Derived class for Multiple Inheritance
class TechLead : public Employee, public ProjectManager, public TeamLead
{
public:
    TechLead(const std::string &empName, int empId, const std::string &project, int teamSize)
        : Employee(empName, empId), ProjectManager(project), TeamLead(teamSize) {}

    void displayInfo() const
    {
        display();
        manageProject();
        leadTeam();
    }
};

```



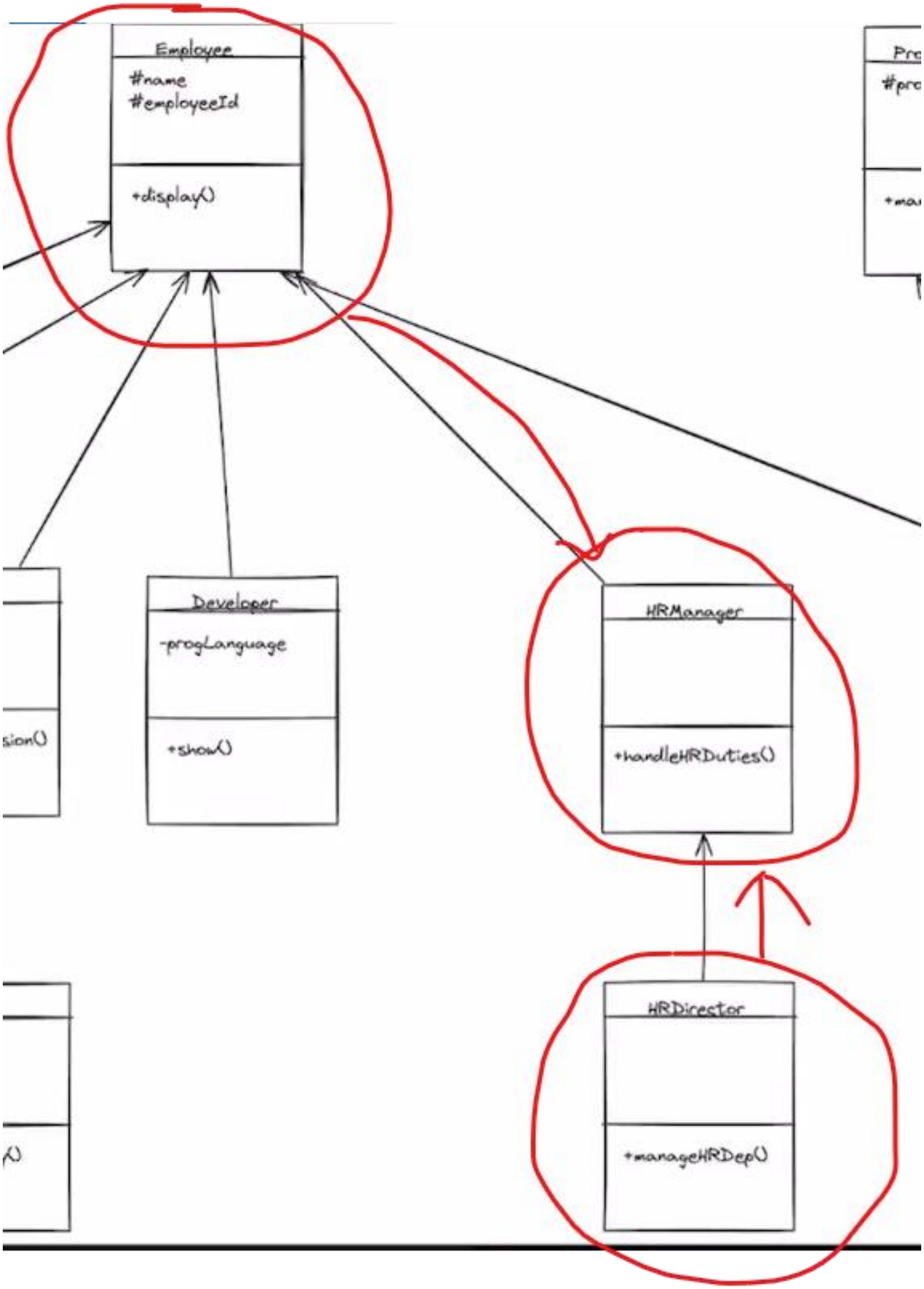
```
int main()
{
    // Single Inheritance
    // Developer dev("Ramu Kaka", 101, "C++");
    // dev.show();

    // // Multiple Inheritance
    TechLead techLead("Anna Dev", 202, "Project X", 5);
    techLead.displayInfo();

```

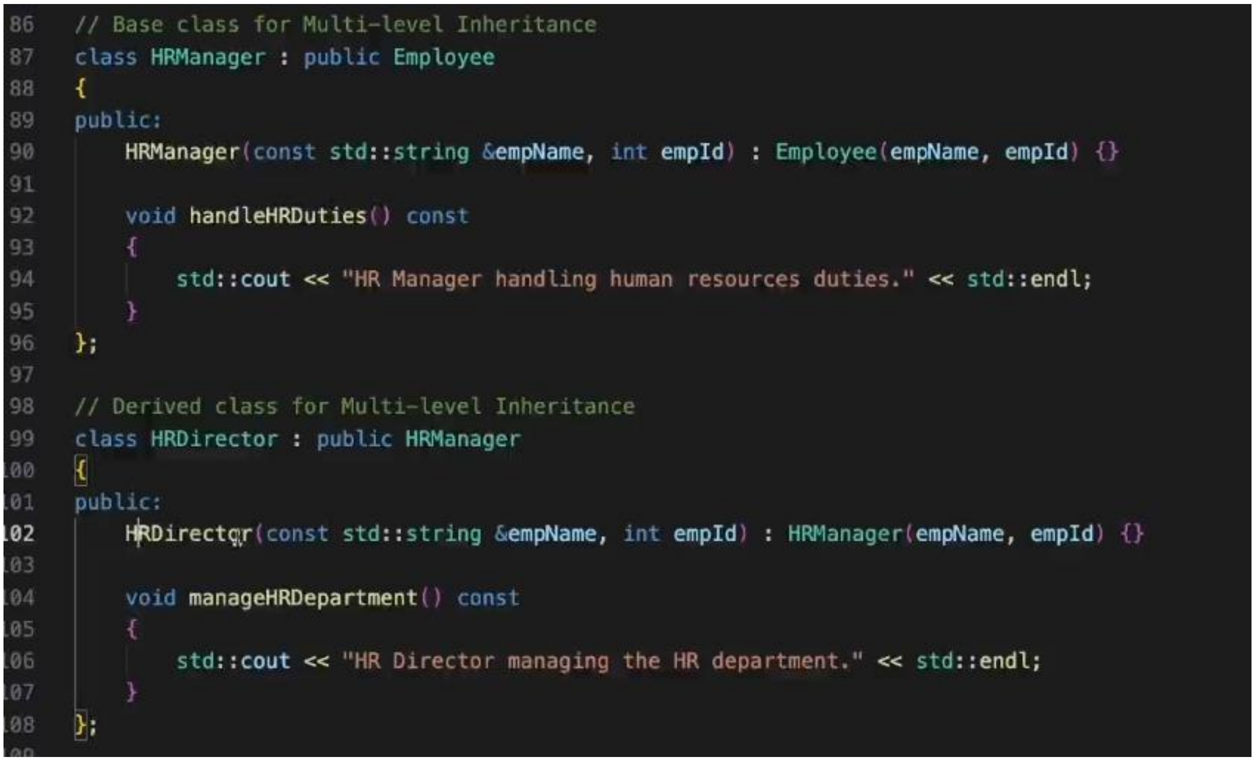
Output



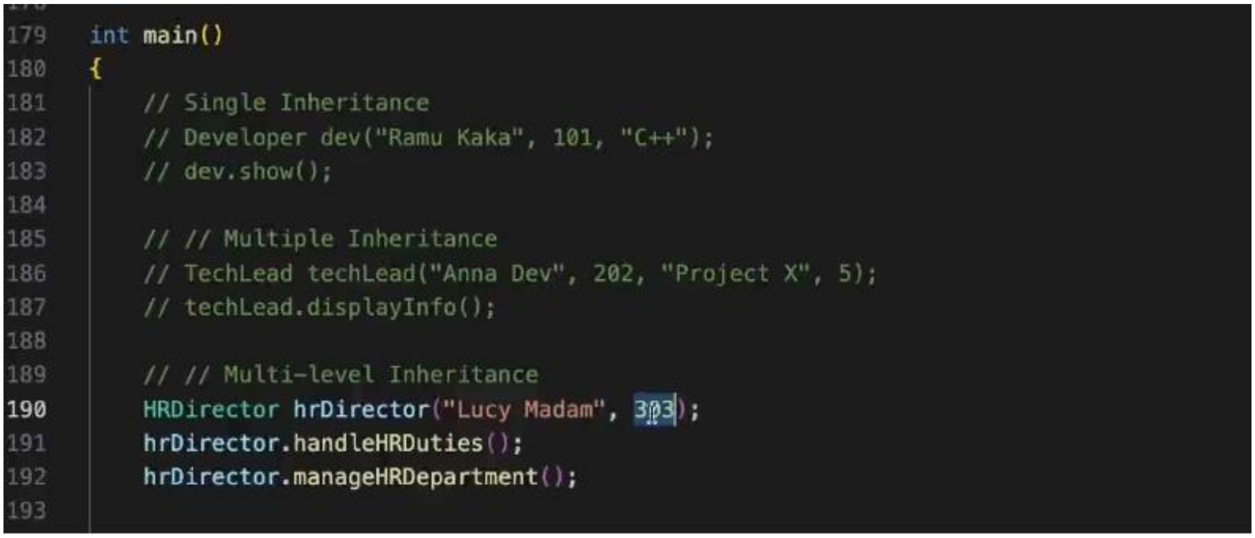
```
(base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop/codehelp/Supra-LLD/C++ Codes/output"
./"TypesOfInheritance"
(base) lakshaykumar@Lakshays-MacBook-Air output % ./"TypesOfInheritance"
Employee: Anna Dev, ID: 202
Project Manager managing project: Project X
Team Lead leading a team of 5 members.
(base) lakshaykumar@Lakshays-MacBook-Air output %
```

Multilevel inheritance

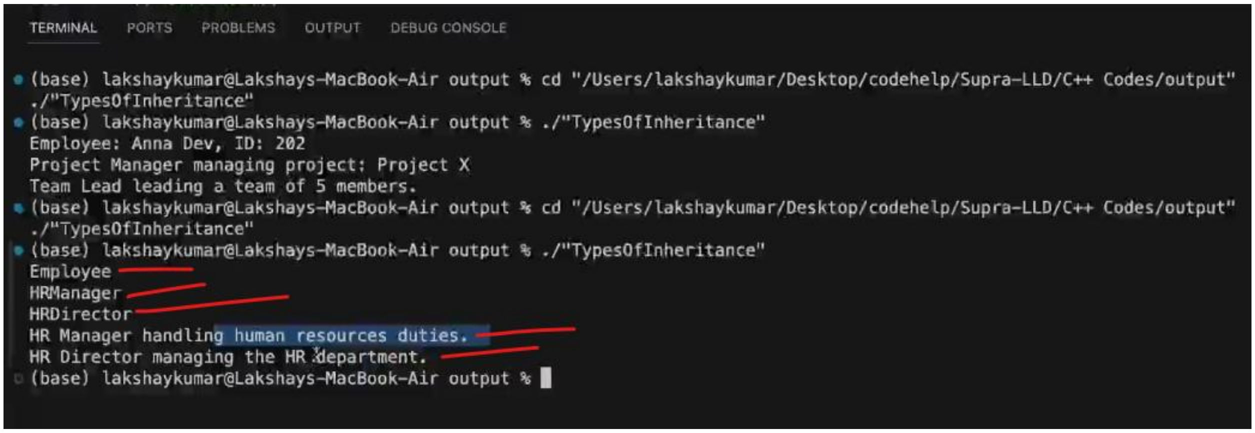




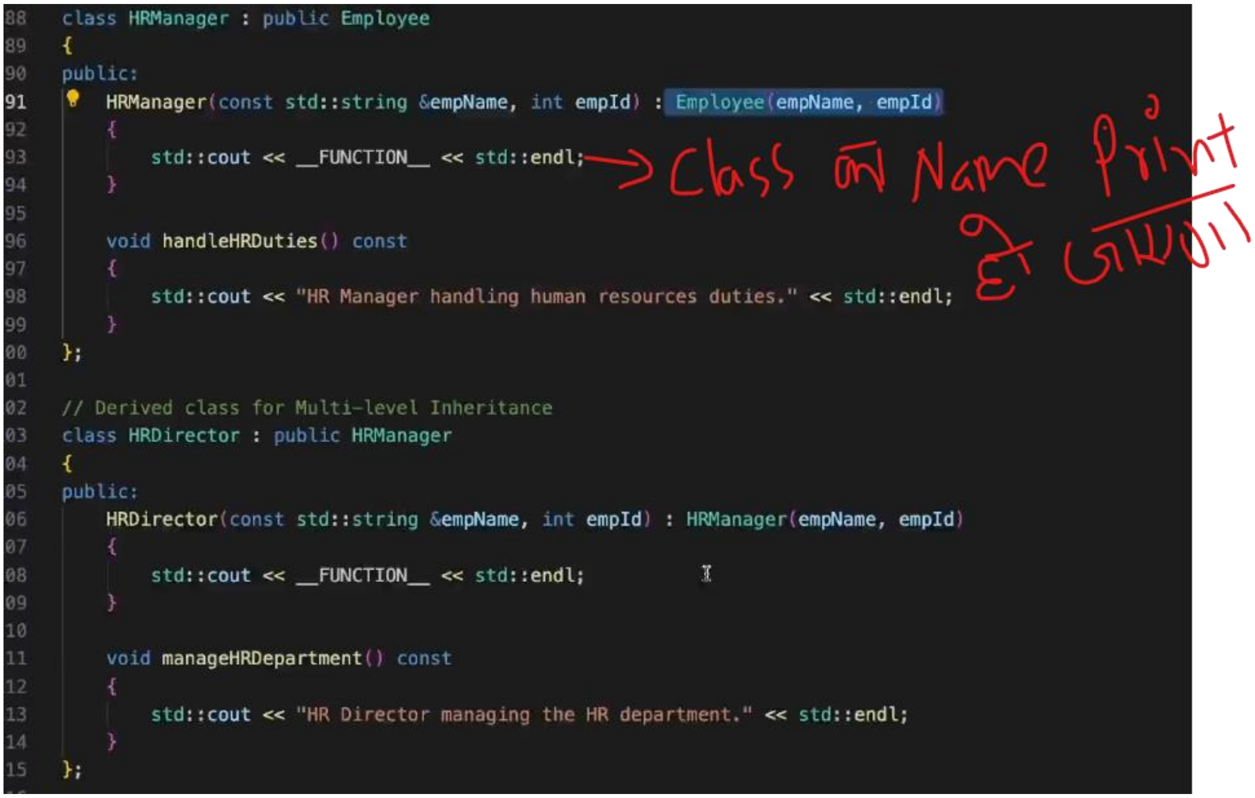
```cpp
// Base class for Multi-level Inheritance
class HRManager : public Employee
{
public:
    HRManager(const std::string &empName, int empId) : Employee(empName, empId) {}

    void handleHRDuties() const
    {
        std::cout << "HR Manager handling human resources duties." << std::endl;
    }
};

// Derived class for Multi-level Inheritance
class HRDirector : public HRManager
{
public:
    HRDirector(const std::string &empName, int empId) : HRManager(empName, empId) {}

    void manageHRDepartment() const
    {
        std::cout << "HR Director managing the HR department." << std::endl;
    }
};
```



```cpp
int main()
{
    // Single Inheritance
    // Developer dev("Ramu Kaka", 101, "C++");
    // dev.show();

    // // Multiple Inheritance
    // TechLead techLead("Anna Dev", 202, "Project X", 5);
    // techLead.displayInfo();

    // // Multi-level Inheritance
    HRDirector hrDirector("Lucy Madam", 303);
    hrDirector.handleHRDuties();
    hrDirector.manageHRDepartment();

```



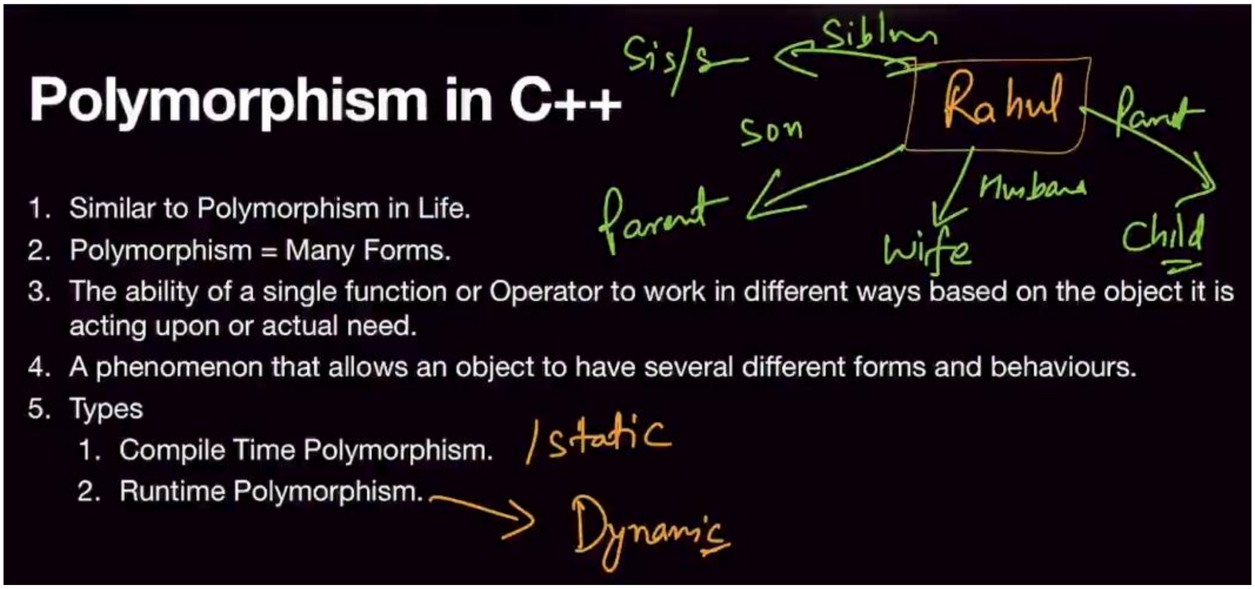
```
● (base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop/codehelp/Supra-LLD/C++ Codes/output"
./"TypesOfInheritance"
● (base) lakshaykumar@Lakshays-MacBook-Air output % ./"TypesOfInheritance"
Employee: Anna Dev, ID: 202
Project Manager managing project: Project X
Team Lead leading a team of 5 members.
● (base) lakshaykumar@Lakshays-MacBook-Air output % cd "/Users/lakshaykumar/Desktop/codehelp/Supra-LLD/C++ Codes/output"
./"TypesOfInheritance"
● (base) lakshaykumar@Lakshays-MacBook-Air output % ./"TypesOfInheritance"
Employee
HRManager
HRDirector
HR Manager handling human resources duties.
HR Director managing the HR department.
○ (base) lakshaykumar@Lakshays-MacBook-Air output % 
```



```cpp
class HRManager : public Employee
{
public:
    HRManager(const std::string &empName, int empId) : Employee(empName, empId)
    {
        std::cout << __FUNCTION__ << std::endl;
    }

    void handleHRDuties() const
    {
        std::cout << "HR Manager handling human resources duties." << std::endl;
    }
};

// Derived class for Multi-level Inheritance
class HRDirector : public HRManager
{
public:
    HRDirector(const std::string &empName, int empId) : HRManager(empName, empId)
    {
        std::cout << __FUNCTION__ << std::endl;
    }

    void manageHRDepartment() const
    {
        std::cout << "HR Director managing the HR department." << std::endl;
    }
};
```

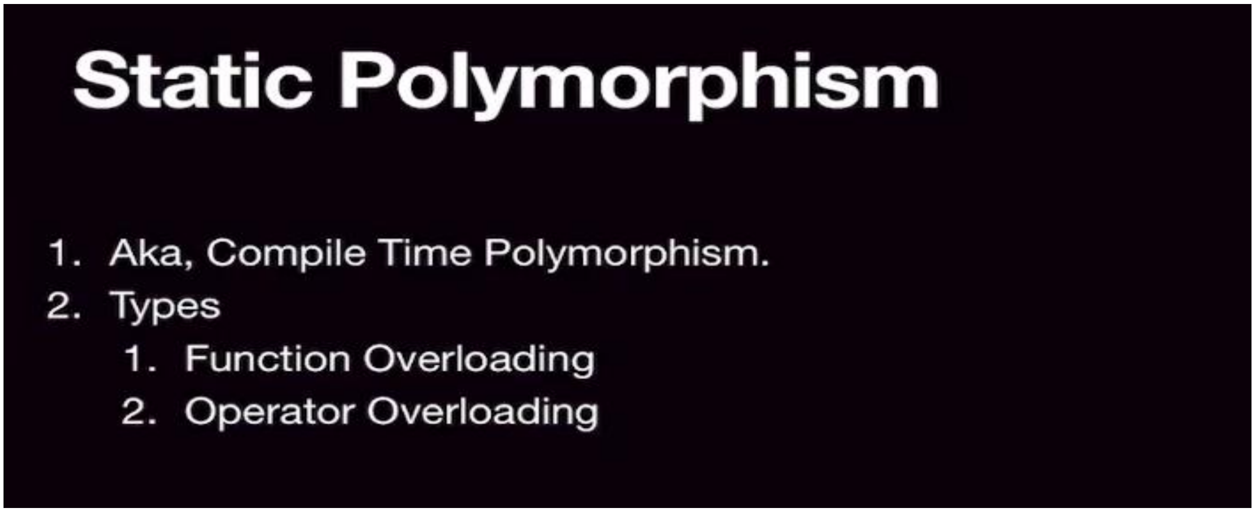
Class का Name Print हो जायेगा।

7.Polymorphism in C++



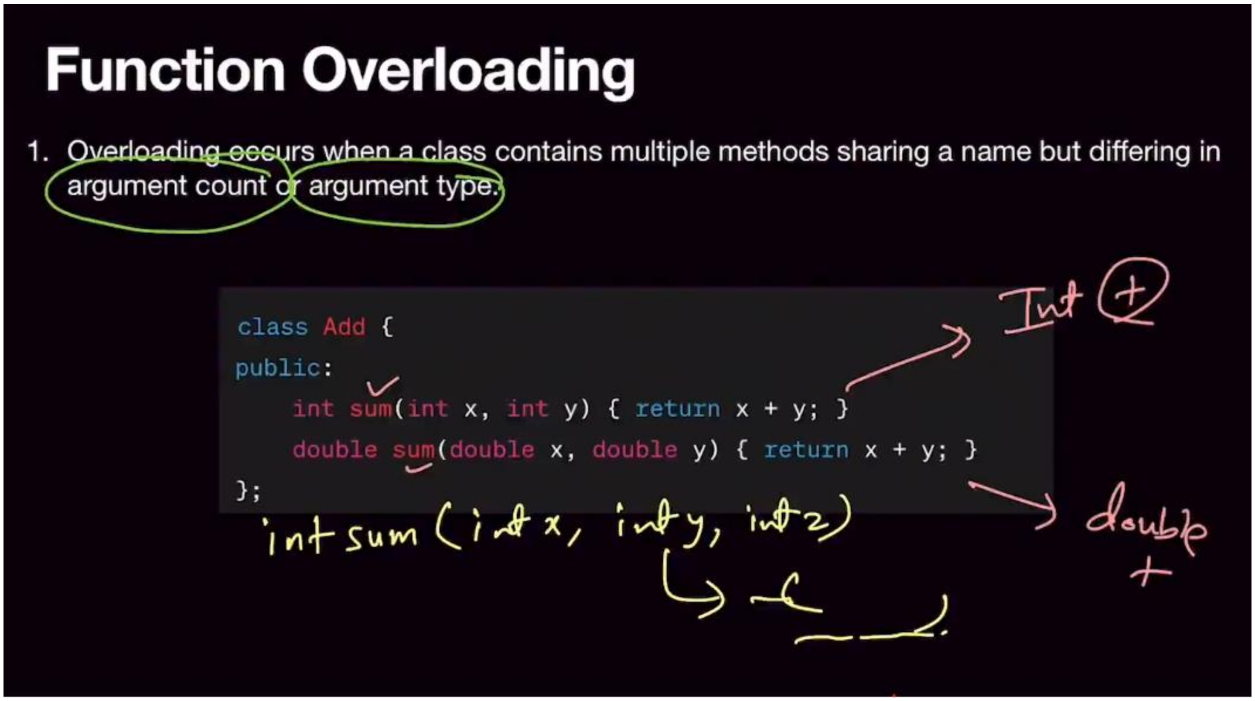
Polymorphism in C++

1. Similar to Polymorphism in Life.
2. Polymorphism = Many Forms.
3. The ability of a single function or Operator to work in different ways based on the object it is acting upon or actual need.
4. A phenomenon that allows an object to have several different forms and behaviours.
5. Types
   1. Compile Time Polymorphism. /static
   2. Runtime Polymorphism. → Dynamic



Static Polymorphism

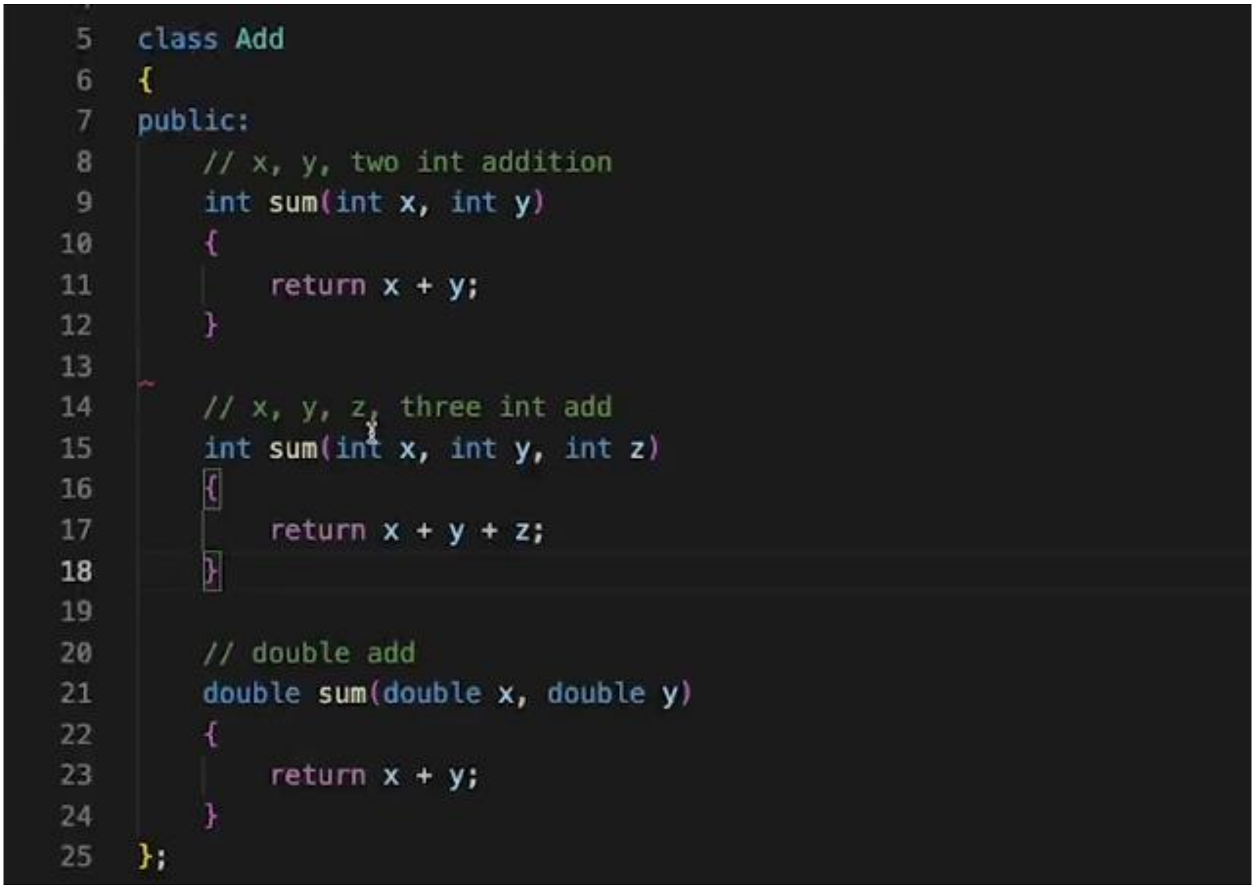
1. Aka, Compile Time Polymorphism.
2. Types
   1. Function Overloading
   2. Operator Overloading



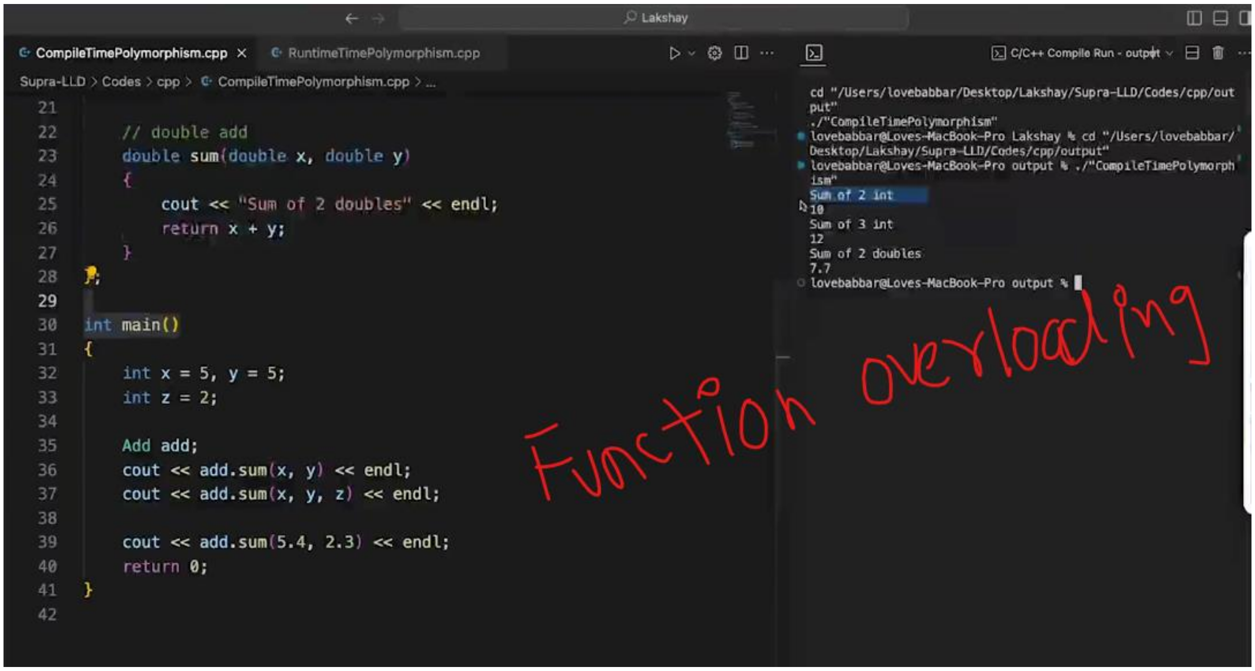
Function Overloading

1. Overloading occurs when a class contains multiple methods sharing a name but differing in argument count or argument type.

```cpp
class Add {
public:
    int sum(int x, int y) { return x + y; }
    double sum(double x, double y) { return x + y; }
};
```



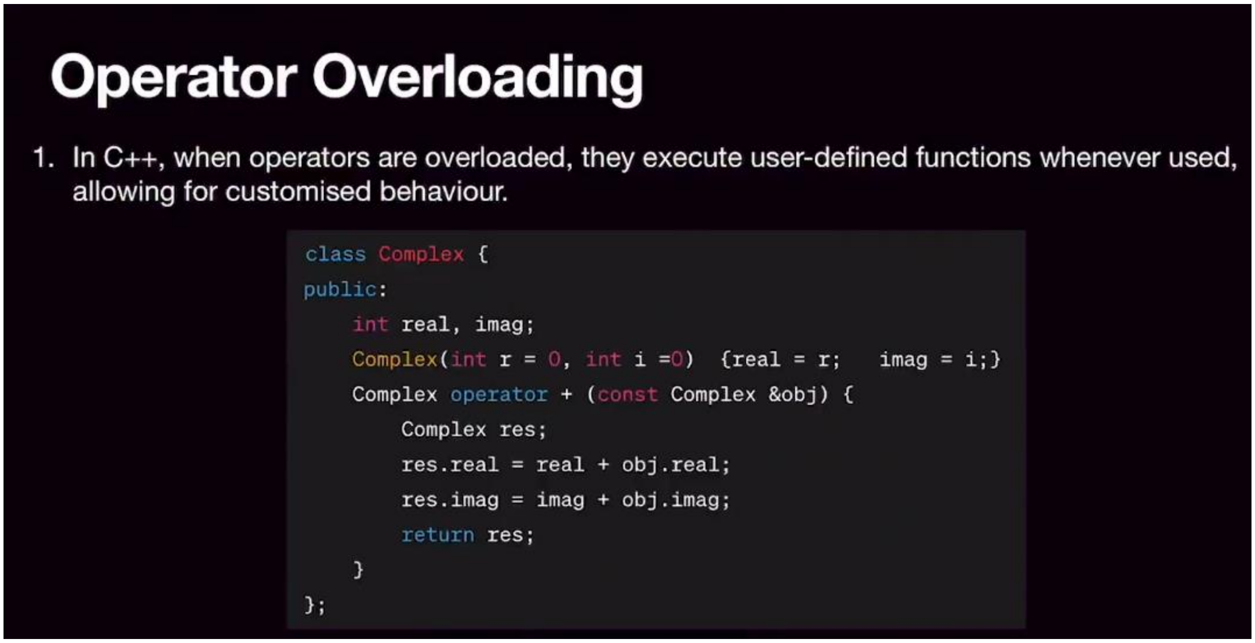
```cpp
class Add
{
public:
    // x, y, two int addition
    int sum(int x, int y)
    {
        return x + y;
    }

    // x, y, z, three int add
    int sum(int x, int y, int z)
    {
        return x + y + z;
    }

    // double add
    double sum(double x, double y)
    {
        return x + y;
    }
};
```


Lakshay
CompileTimePolymorphism.cpp
RuntimeTimePolymorphism.cpp
C/C++ Compile Run - output
Supra-LLD > Codes > cpp > CompileTimePolymorphism.cpp > ...

```cpp

    // double add
    double sum(double x, double y)
    {
        cout << "Sum of 2 doubles" << endl;
        return x + y;
    }
};

int main()
{
    int x = 5, y = 5;
    int z = 2;

    Add add;
    cout << add.sum(x, y) << endl;
    cout << add.sum(x, y, z) << endl;

    cout << add.sum(5.4, 2.3) << endl;
    return 0;
}

```

```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"
Sum of 2 int
10
Sum of 3 int
12
Sum of 2 doubles
7.7
lovebabbar@Loves-MacBook-Pro output %
```

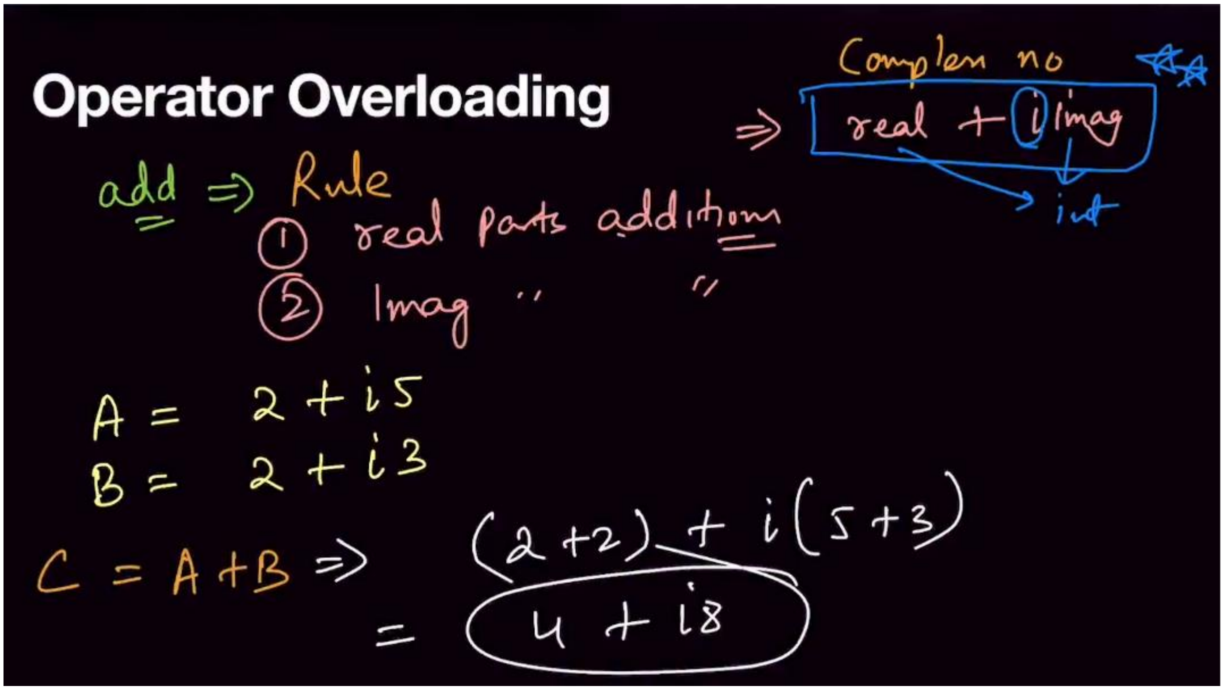
Function overloading



Operator Overloading

1. In C++, when operators are overloaded, they execute user-defined functions whenever used, allowing for customised behaviour.

```cpp
class Complex {
public:
    int real, imag;
    Complex(int r = 0, int i =0)  {real = r;   imag = i;}
    Complex operator + (const Complex &obj) {
        Complex res;
        res.real = real + obj.real;
        res.imag = imag + obj.imag;
        return res;
    }
};
```



Operator Overloading

Complex no

⇒ real + i imag → int

add ⇒ Rule

① real parts addition

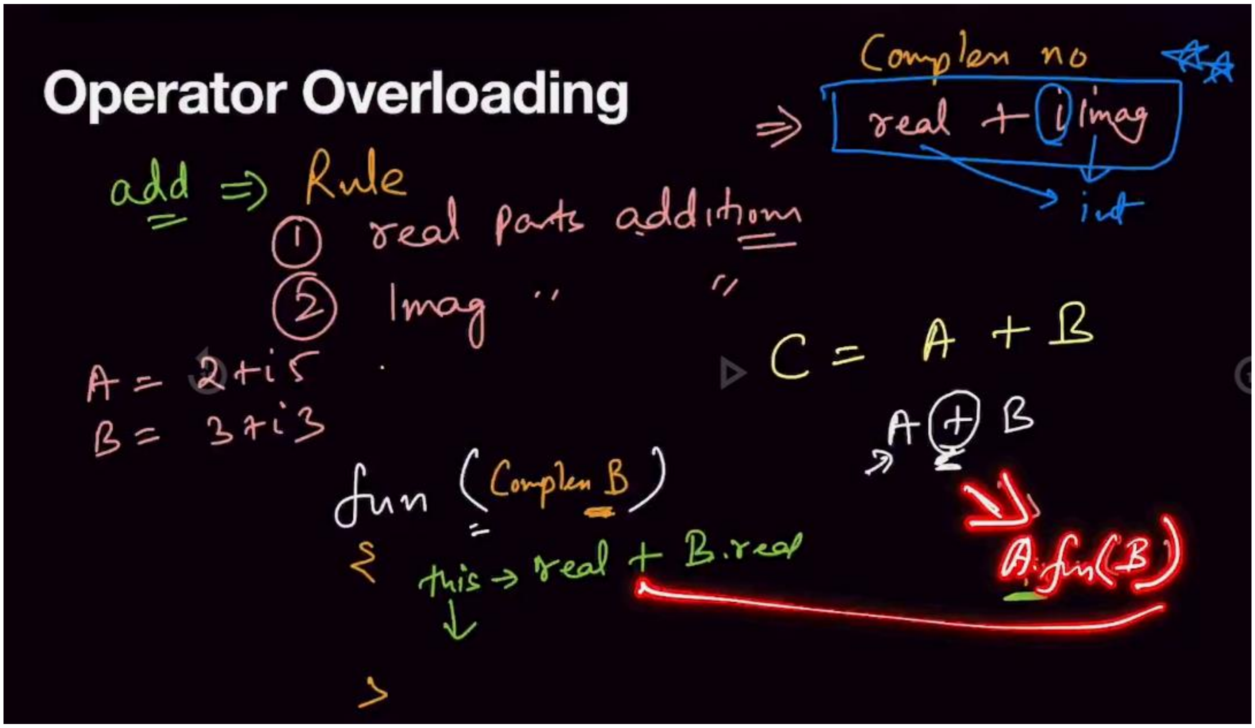
② Imag " "

$A = 2 + i5$

$B = 2 + i3$

$C = A + B \Rightarrow (2+2) + i(5+3)$

$= \boxed{4 + i8}$



Operator Overloading

Complex no

⇒ real + i imag → int

add ⇒ Rule

① real parts addition

② Imag " "

$A = 2+i5$

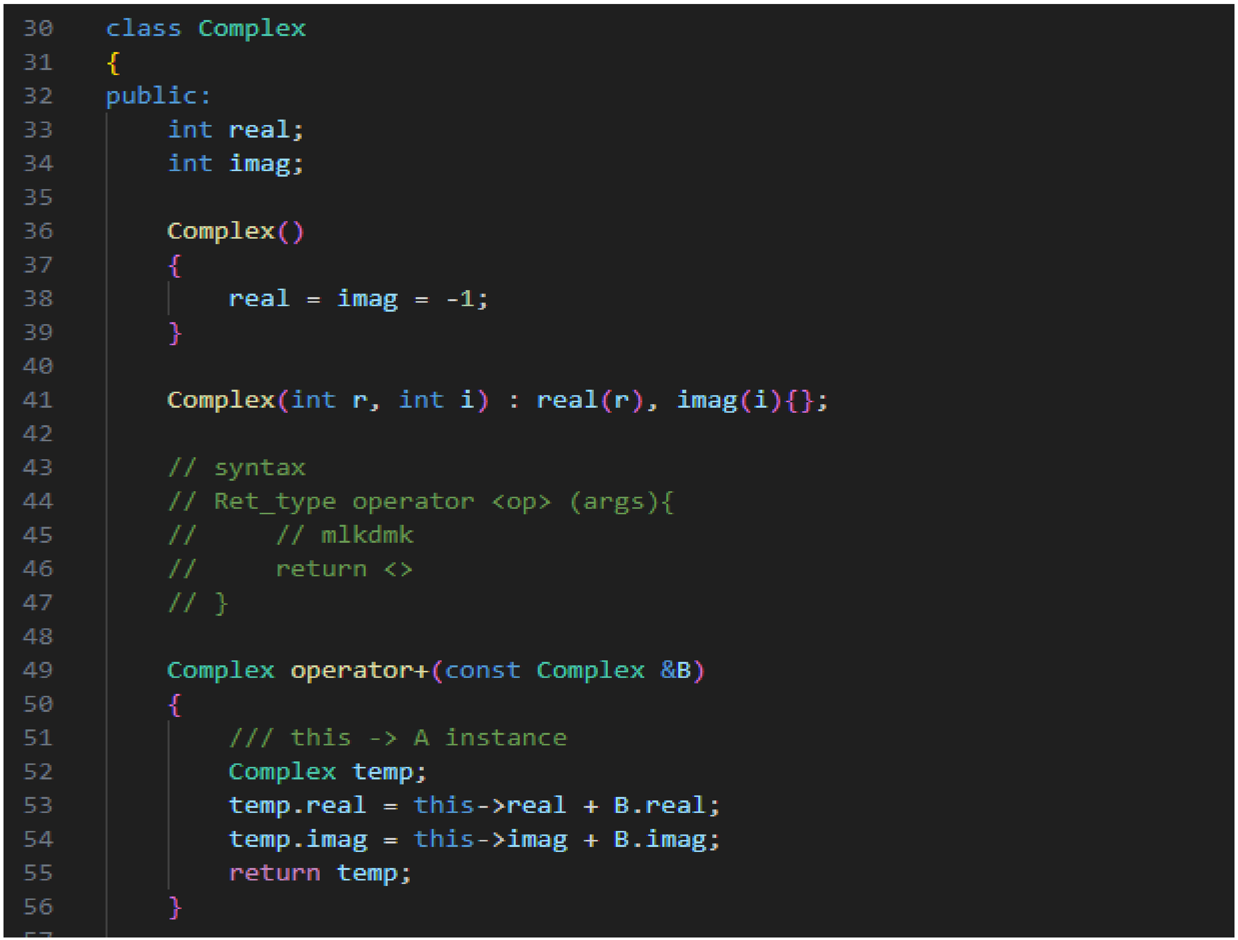
$B = 3+i3$

$C = A + B$

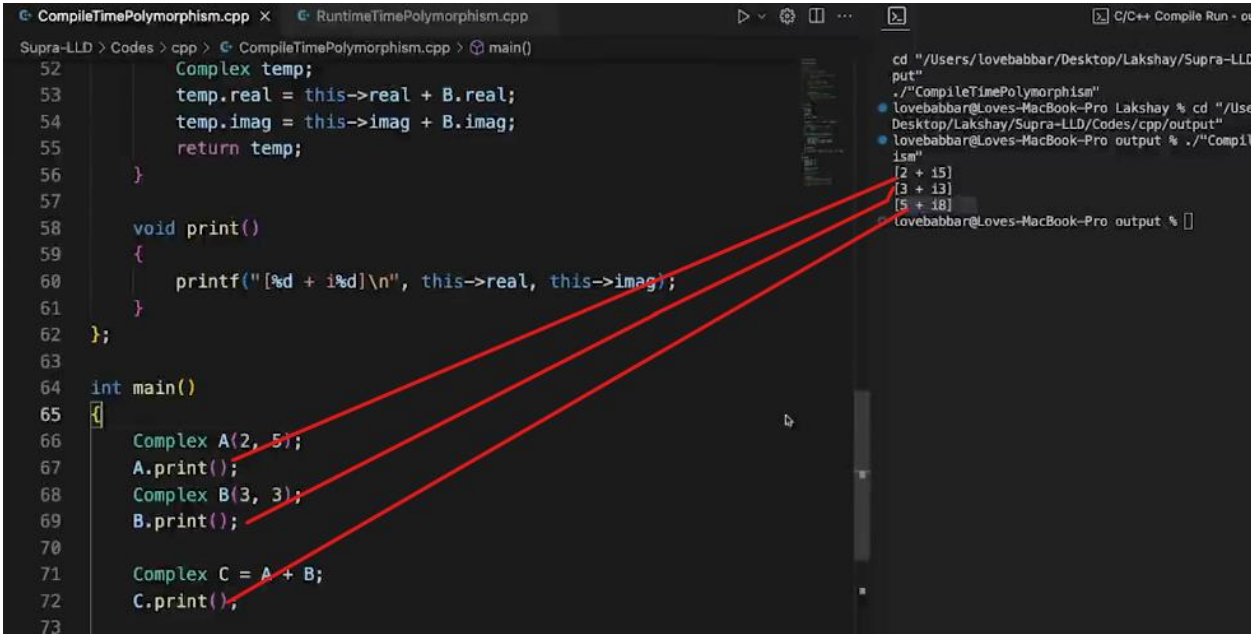
$A \oplus B$

A.fun(B)

```
fun (Complex B)
{
    this -> real + B.real
    ↓
}
```



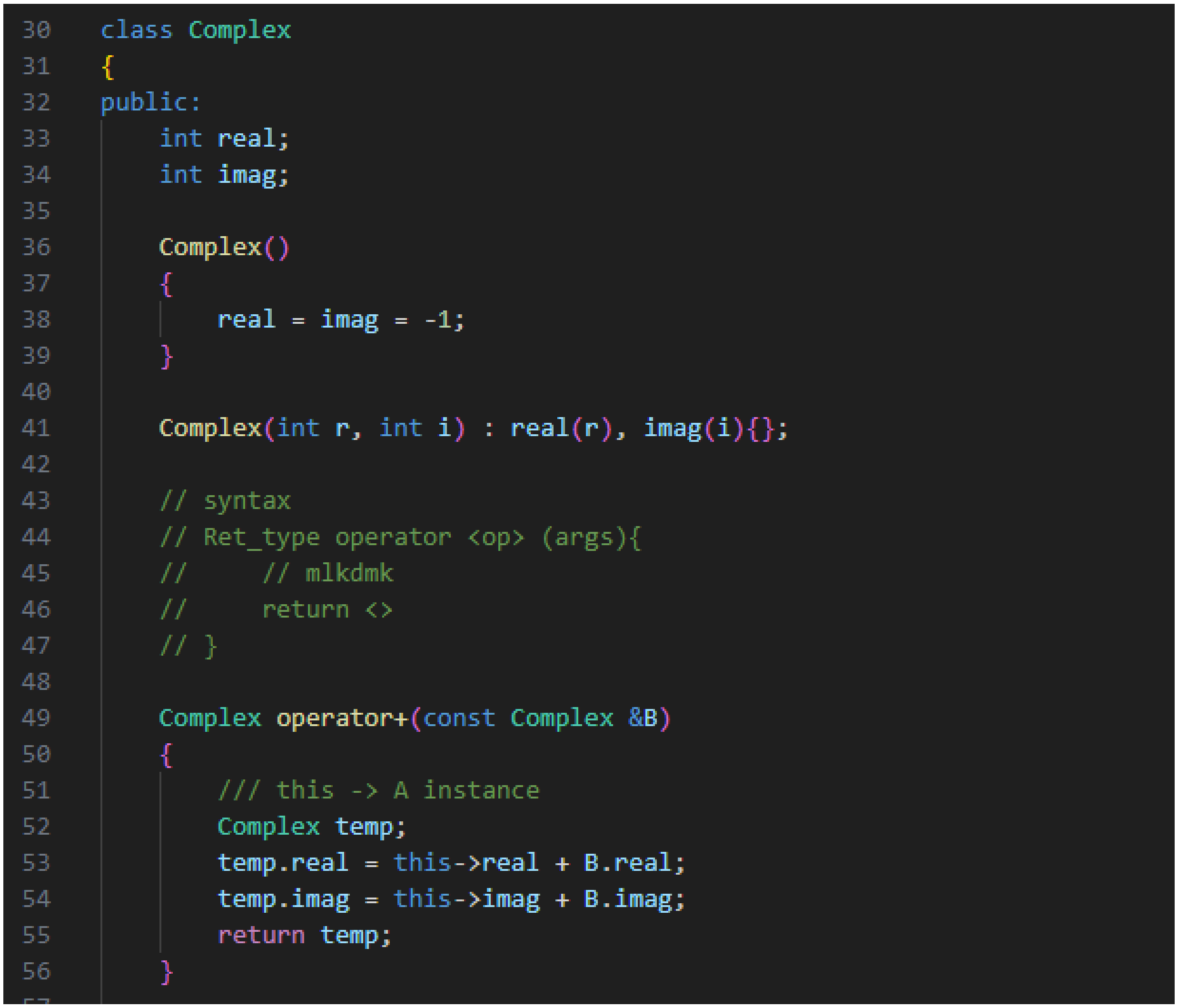
```cpp
class Complex
{
public:
    int real;
    int imag;

    Complex()
    {
        real = imag = -1;
    }

    Complex(int r, int i) : real(r), imag(i){};

    // syntax
    // Ret_type operator <op> (args){
    //      // mlkdmk
    //      return <>
    // }

    Complex operator+(const Complex &B)
    {
        /// this -> A instance
        Complex temp;
        temp.real = this->real + B.real;
        temp.imag = this->imag + B.imag;
        return temp;
    }
```



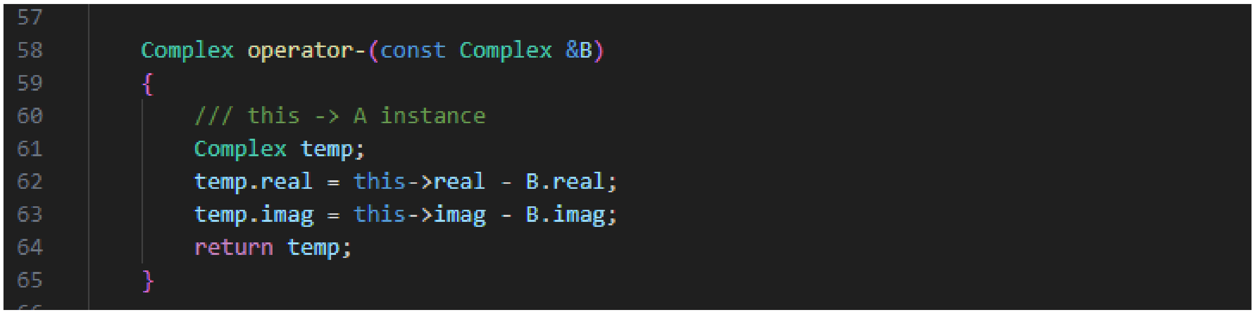
CompileTimePolymorphism.cpp ×
RuntimeTimePolymorphism.cpp
Supra-LLD › Codes › cpp › CompileTimePolymorphism.cpp › main()
C/C++ Compile Run - ou

```cpp
        Complex temp;
        temp.real = this->real + B.real;
        temp.imag = this->imag + B.imag;
        return temp;
    }

    void print()
    {
        printf("[%d + i%d]\n", this->real, this->imag);
    }
};

int main()
{
    Complex A(2, 5);
    A.print();
    Complex B(3, 3);
    B.print();

    Complex C = A + B;
    C.print();
```

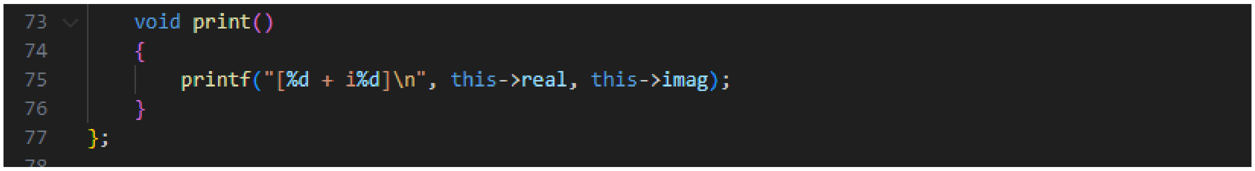
```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD
put"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Use
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"Compil
ism"
[2 + i5]
[3 + i3]
[5 + i8]
lovebabbar@Loves-MacBook-Pro output % 
```



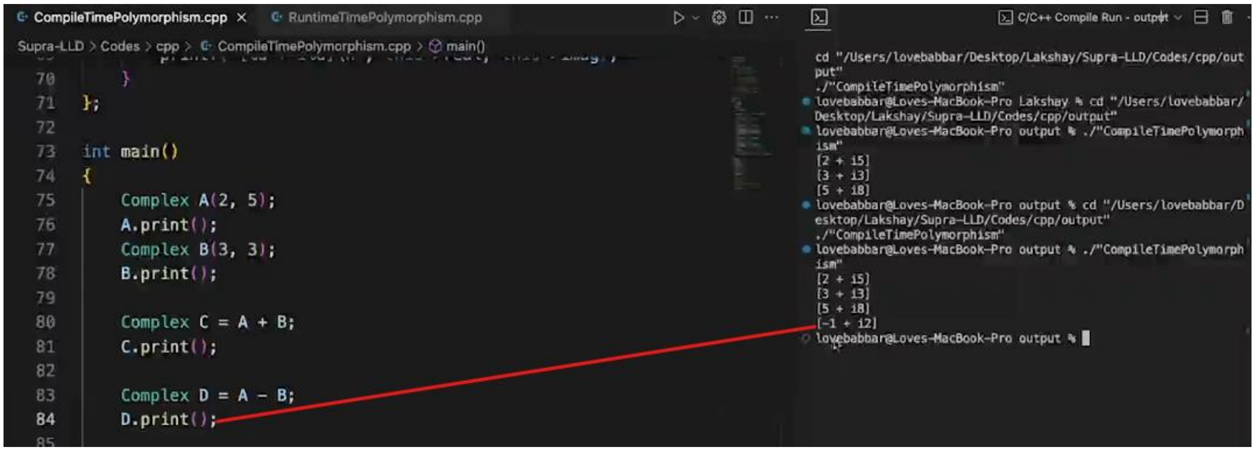
```cpp
class Complex
{
public:
    int real;
    int imag;

    Complex()
    {
        real = imag = -1;
    }

    Complex(int r, int i) : real(r), imag(i){};

    // syntax
    // Ret_type operator <op> (args){
    //     // mlkdmk
    //     return <>
    // }

    Complex operator+(const Complex &B)
    {
        /// this -> A instance
        Complex temp;
        temp.real = this->real + B.real;
        temp.imag = this->imag + B.imag;
        return temp;
    }
```



```cpp

    Complex operator-(const Complex &B)
    {
        /// this -> A instance
        Complex temp;
        temp.real = this->real - B.real;
        temp.imag = this->imag - B.imag;
        return temp;
    }
```



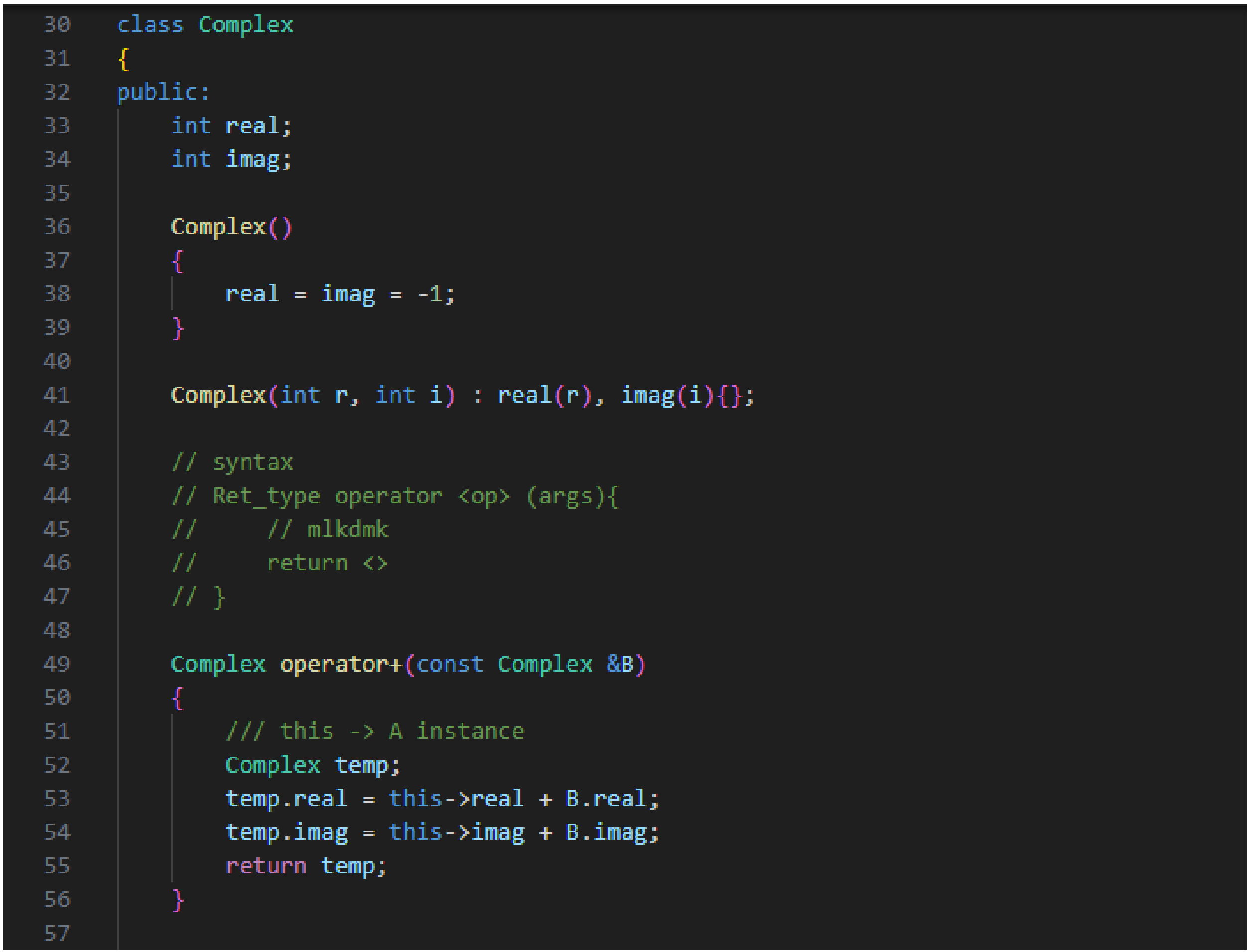
```cpp
    void print()
    {
        printf("[%d + i%d]\n", this->real, this->imag);
    }
};
```



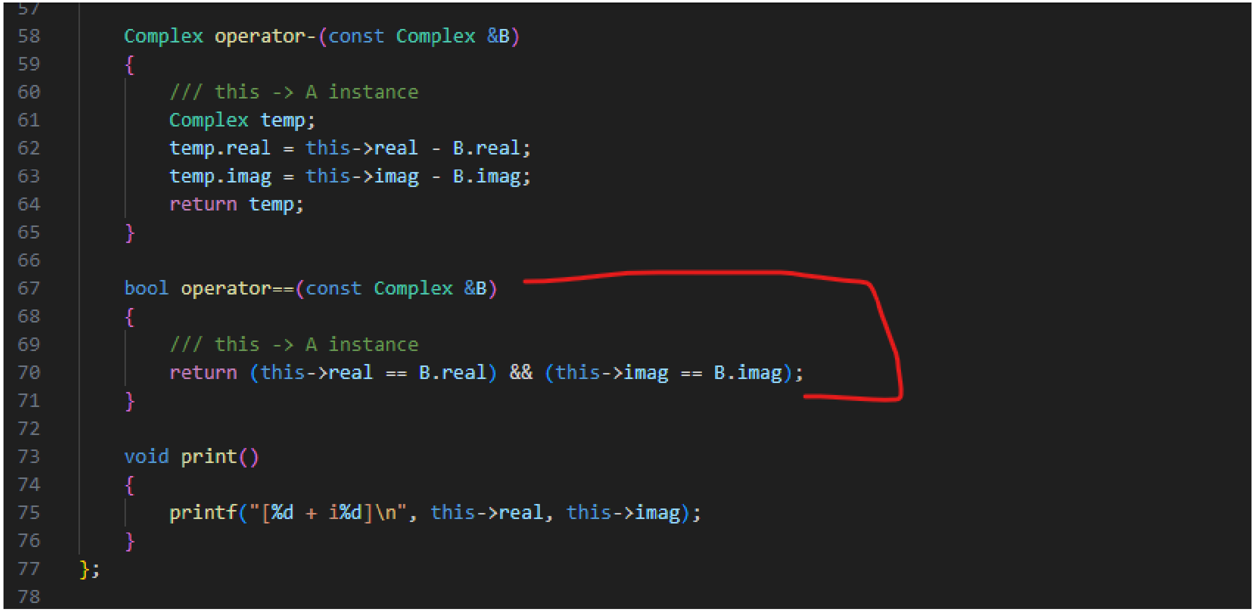
CompileTimePolymorphism.cpp
RuntimeTimePolymorphism.cpp
C/C++ Compile Run - output
Supra-LLD > Codes > cpp > CompileTimePolymorphism.cpp > main()

```cpp
        printf("[%d + i%d]\n", this->real, this->imag);
    }
};

int main()
{
    Complex A(2, 5);
    A.print();
    Complex B(3, 3);
    B.print();

    Complex C = A + B;
    C.print();

    Complex D = A - B;
    D.print();
```

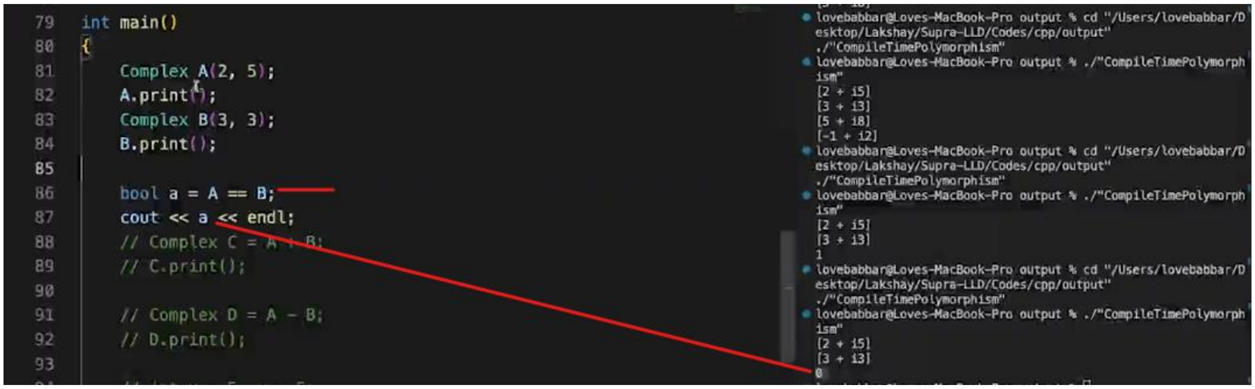
```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/out
put"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lovebabbar/
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorph
ism"
[2 + i5]
[3 + i3]
[5 + i8]
lovebabbar@Loves-MacBook-Pro output % cd "/Users/lovebabbar/D
esktop/Lakshay/Supra-LLD/Codes/cpp/output"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorph
ism"
[2 + i5]
[3 + i3]
[5 + i8]
[-1 + i2]
lovebabbar@Loves-MacBook-Pro output %
```



```cpp
class Complex
{
public:
    int real;
    int imag;

    Complex()
    {
        real = imag = -1;
    }

    Complex(int r, int i) : real(r), imag(i){};

    // syntax
    // Ret_type operator <op> (args){
    //     // mlkdmk
    //     return <>
    // }

    Complex operator+(const Complex &B)
    {
        /// this -> A instance
        Complex temp;
        temp.real = this->real + B.real;
        temp.imag = this->imag + B.imag;
        return temp;
    }

```

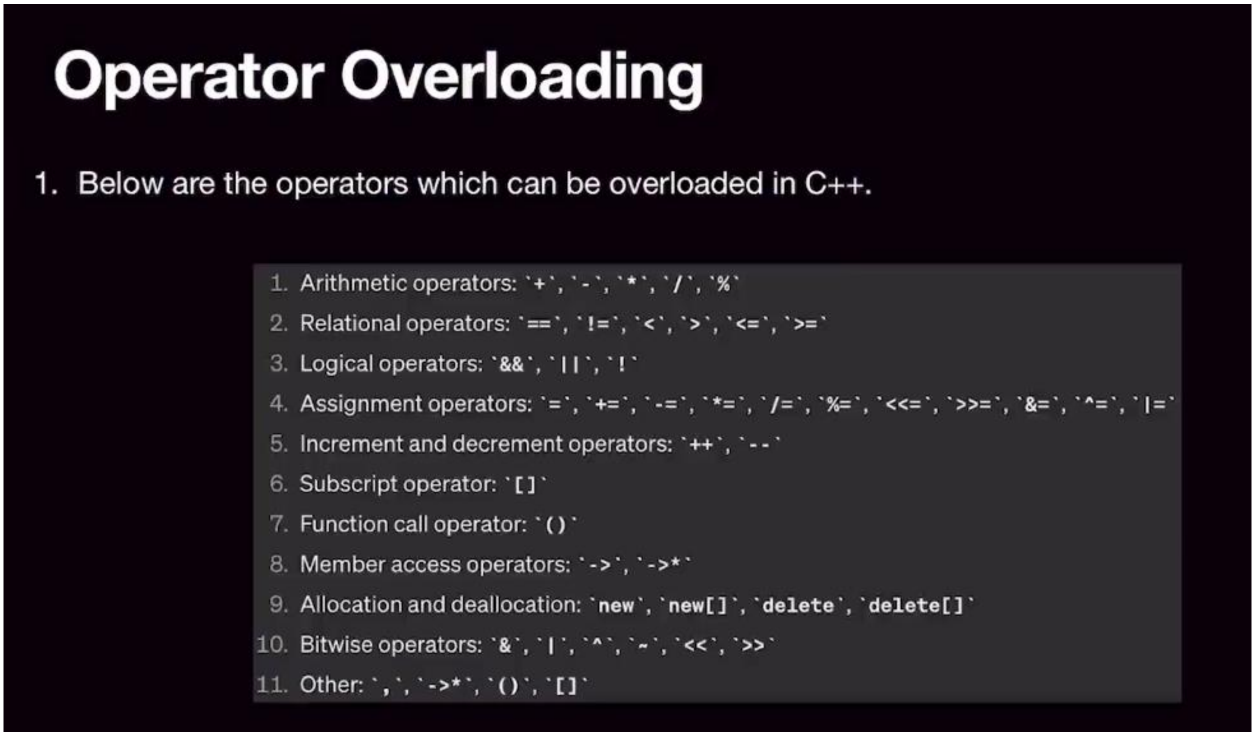


```cpp

    Complex operator-(const Complex &B)
    {
        /// this -> A instance
        Complex temp;
        temp.real = this->real - B.real;
        temp.imag = this->imag - B.imag;
        return temp;
    }

    bool operator==(const Complex &B)
    {
        /// this -> A instance
        return (this->real == B.real) && (this->imag == B.imag);
    }

    void print()
    {
        printf("[%d + i%d]\n", this->real, this->imag);
    }
};

```



```
int main()
{
    Complex A(2, 5);
    A.print();
    Complex B(3, 3);
    B.print();

    bool a = A == B;
    cout << a << endl;
    // Complex C = A + B;
    // C.print();

    // Complex D = A - B;
    // D.print();
```

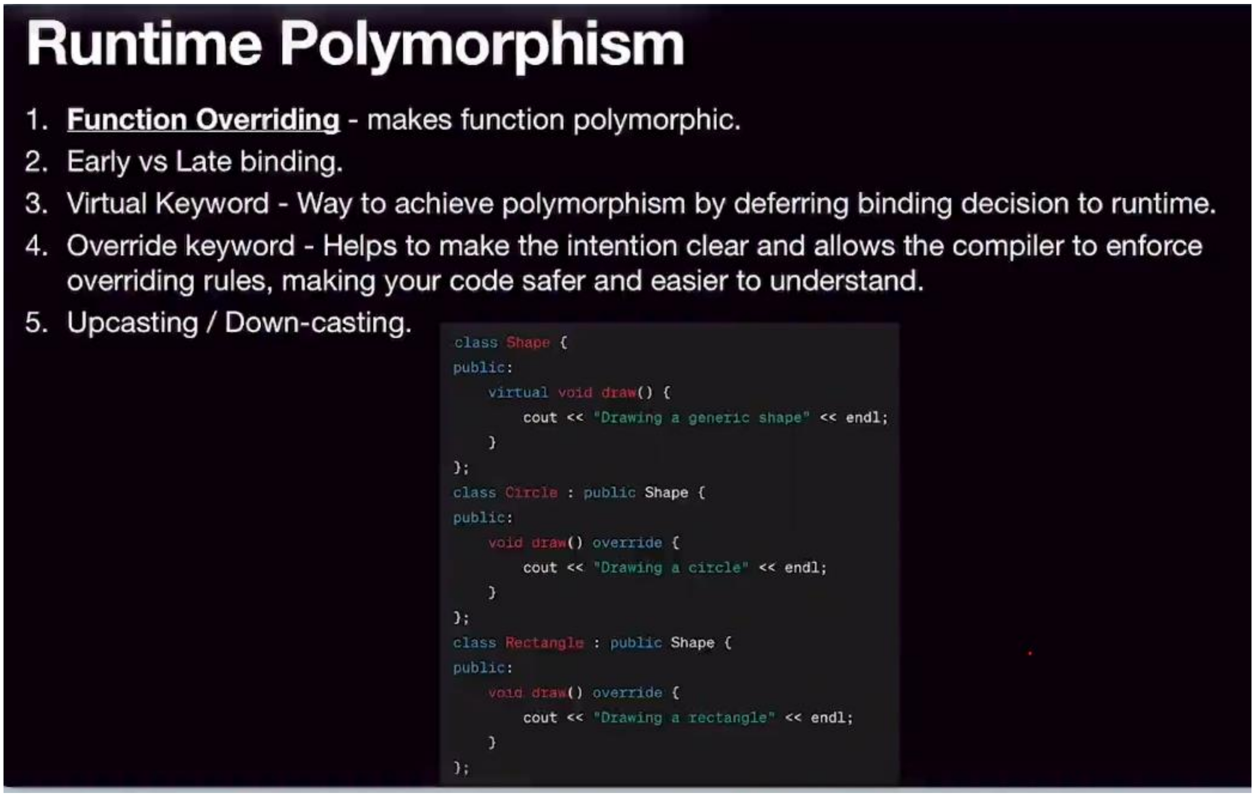
```
lovebabbar@Loves-MacBook-Pro output % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"
[2 + i5]
[3 + i3]
[5 + i8]
[-1 + i2]
lovebabbar@Loves-MacBook-Pro output % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"
[2 + i5]
[3 + i3]
1
lovebabbar@Loves-MacBook-Pro output % cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
./"CompileTimePolymorphism"
lovebabbar@Loves-MacBook-Pro output % ./"CompileTimePolymorphism"
[2 + i5]
[3 + i3]
0
```



Operator Overloading

1. Below are the operators which can be overloaded in C++.

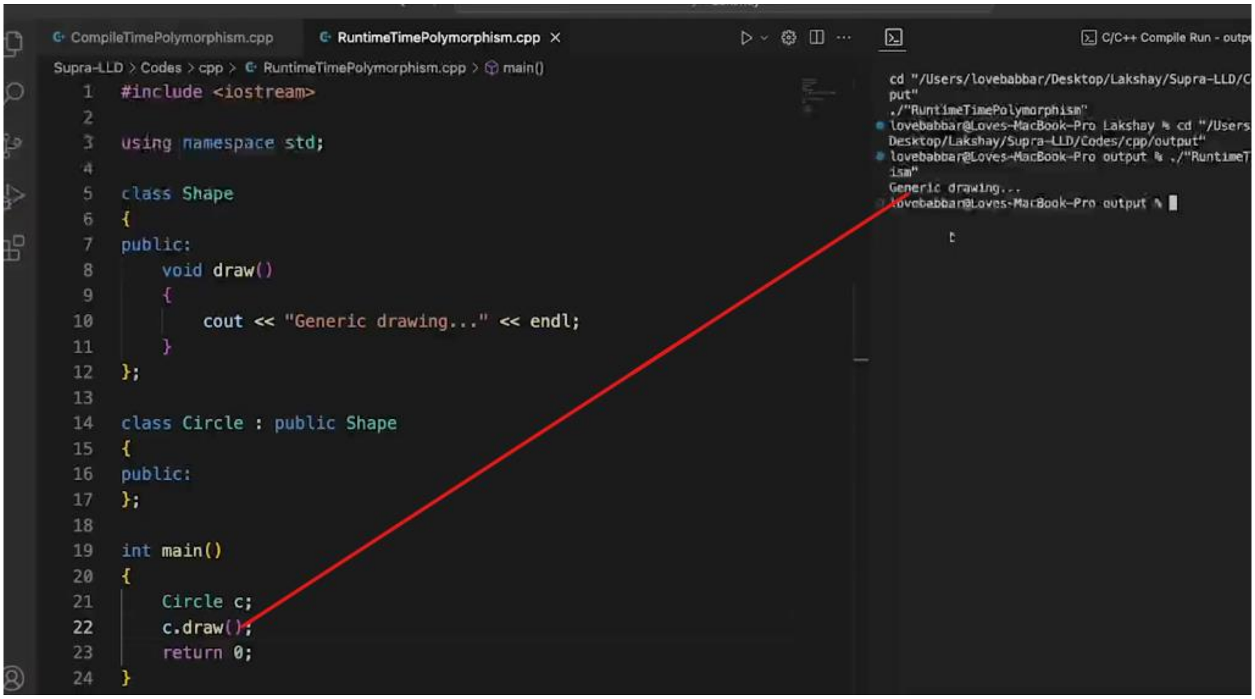
    1. Arithmetic operators: `+`, `-`, `*`, `/`, `%`
    2. Relational operators: `==`, `!=`, `<`, `>`, `<=`, `>=`
    3. Logical operators: `&&`, `||`, `!`
    4. Assignment operators: `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `<<=`, `>>=`, `&=`, `^=`, `|=`
    5. Increment and decrement operators: `++`, `--`
    6. Subscript operator: `[]`
    7. Function call operator: `()`
    8. Member access operators: `->`, `->*`
    9. Allocation and deallocation: `new`, `new[]`, `delete`, `delete[]`
    10. Bitwise operators: `&`, `|`, `^`, `~`, `<<`, `>>`
    11. Other: `,`, `->*`, `()`, `[]`



Runtime Polymorphism

1. **Function Overriding** - makes function polymorphic.
2. Early vs Late binding.
3. Virtual Keyword - Way to achieve polymorphism by deferring binding decision to runtime.
4. Override keyword - Helps to make the intention clear and allows the compiler to enforce overriding rules, making your code safer and easier to understand.
5. Upcasting / Down-casting.

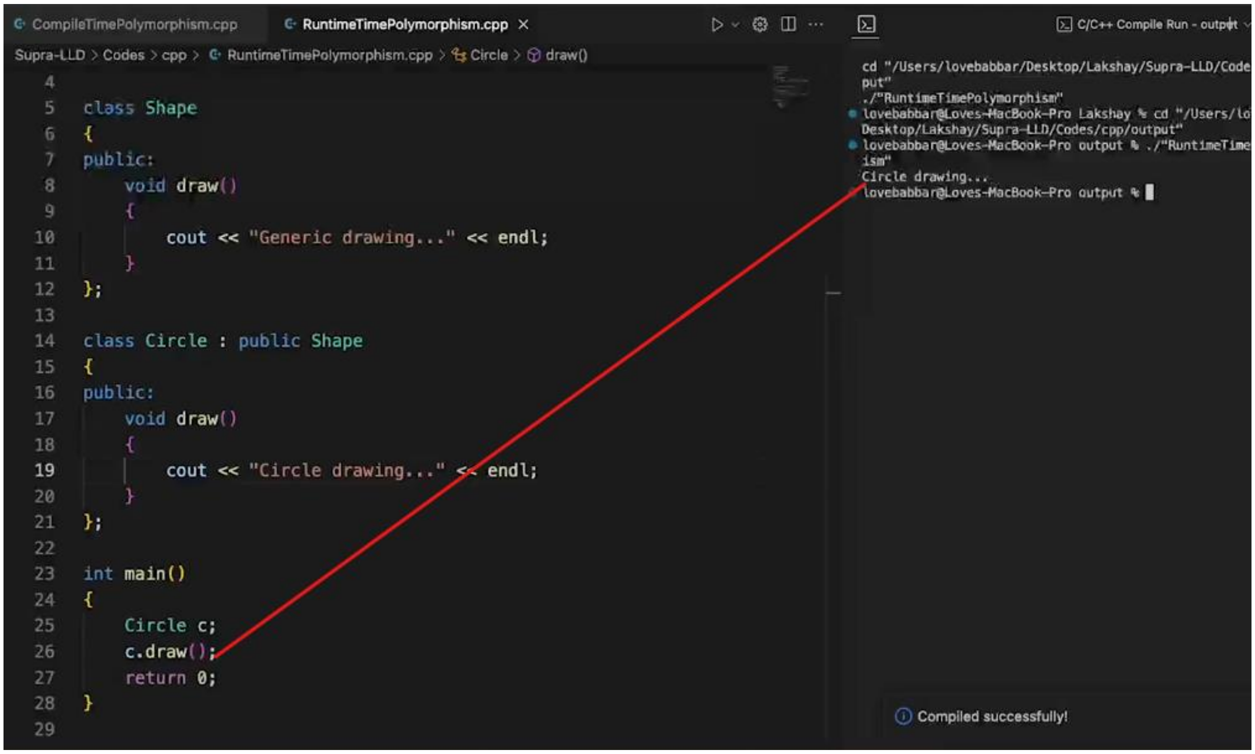
```cpp
class Shape {
public:
    virtual void draw() {
        cout << "Drawing a generic shape" << endl;
    }
};
class Circle : public Shape {
public:
    void draw() override {
        cout << "Drawing a circle" << endl;
    }
};
class Rectangle : public Shape {
public:
    void draw() override {
        cout << "Drawing a rectangle" << endl;
    }
};
```



CompileTimePolymorphism.cpp
RuntimeTimePolymorphism.cpp
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > main()
C/C++ Compile Run - outp

```cpp
#include <iostream>

using namespace std;

class Shape
{
public:
    void draw()
    {
        cout << "Generic drawing..." << endl;
    }
};

class Circle : public Shape
{
public:
};

int main()
{
    Circle c;
    c.draw();
    return 0;
}
```

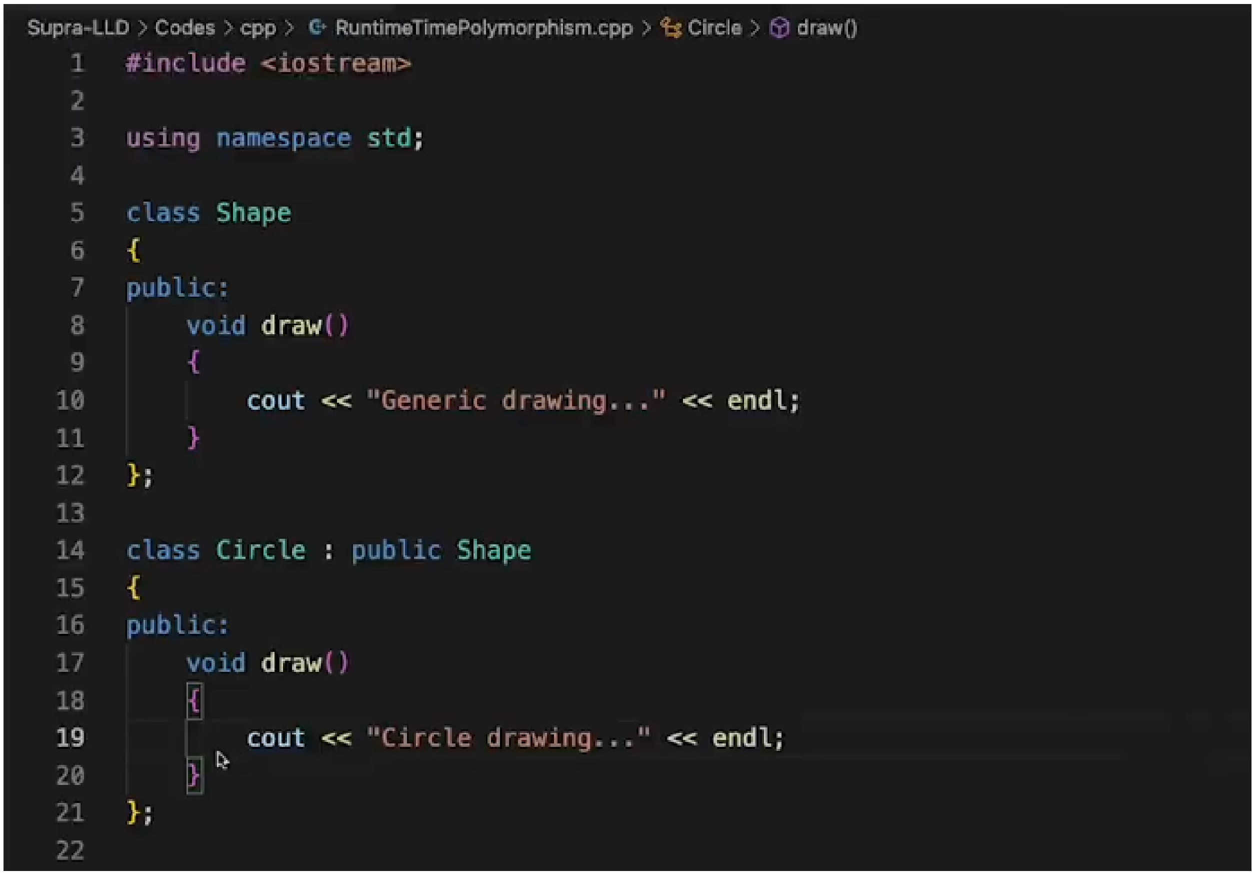
```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/C
put"
./"RuntimeTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeT
ism"
Generic drawing...
lovebabbar@Loves-MacBook-Pro output %
```


CompileTimePolymorphism.cpp
RuntimeTimePolymorphism.cpp
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > Circle > draw()
C/C++ Compile Run - output

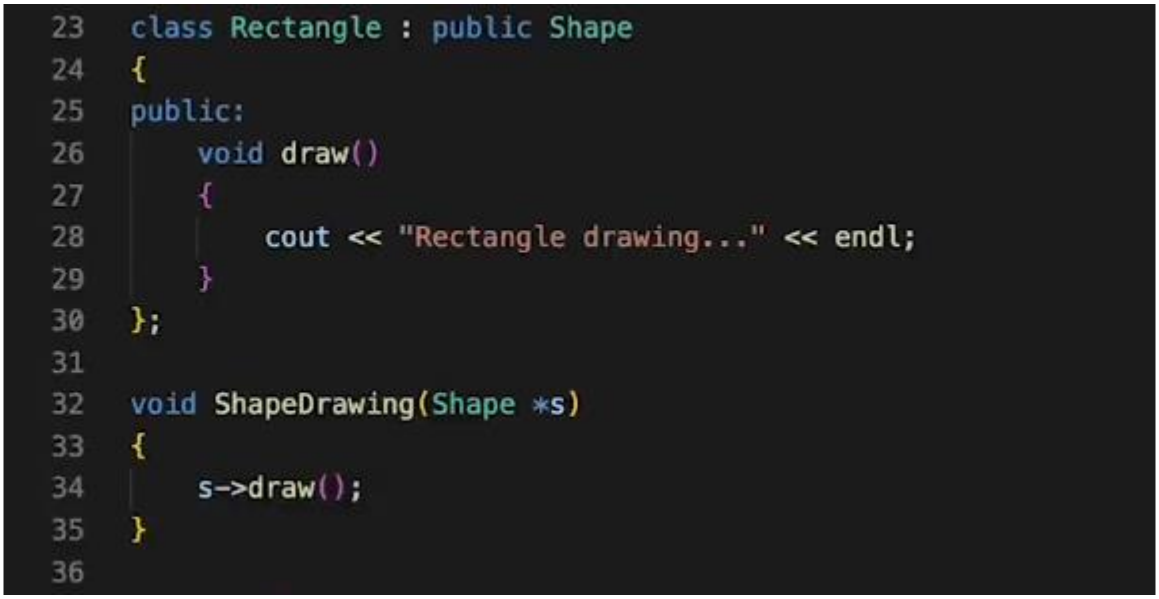
```cpp

class Shape
{
public:
    void draw()
    {
        cout << "Generic drawing..." << endl;
    }
};

class Circle : public Shape
{
public:
    void draw()
    {
        cout << "Circle drawing..." << endl;
    }
};

int main()
{
    Circle c;
    c.draw();
    return 0;
}

```

```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Code
put"
./"RuntimeTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lo
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeTime
ism"
Circle drawing...
lovebabbar@Loves-MacBook-Pro output % 
```

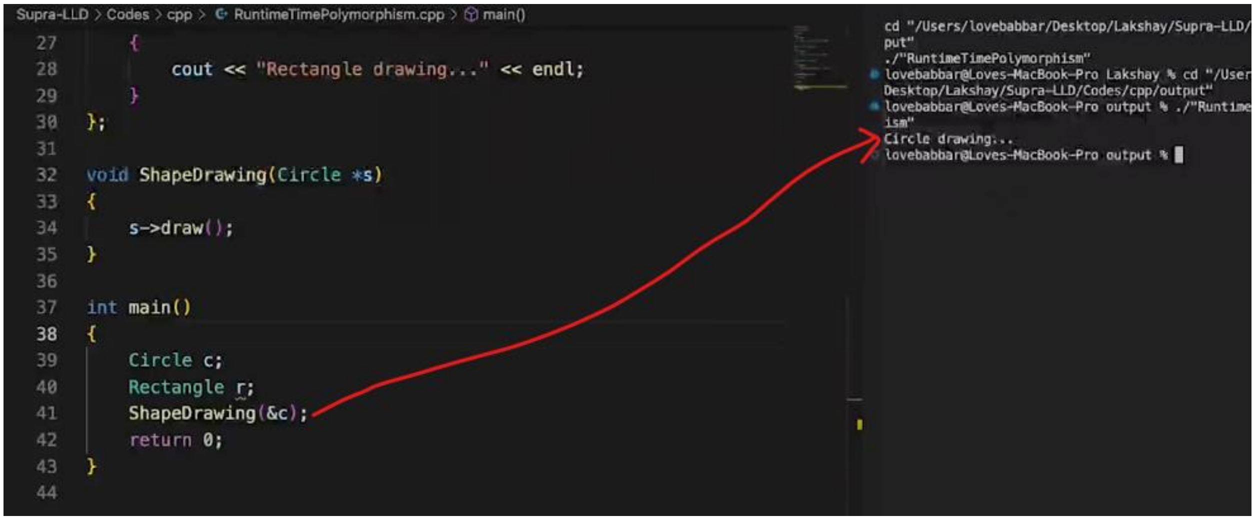
Compiled successfully!


Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > Circle > draw()

```cpp
#include <iostream>

using namespace std;

class Shape
{
public:
    void draw()
    {
        cout << "Generic drawing..." << endl;
    }
};

class Circle : public Shape
{
public:
    void draw()
    {
        cout << "Circle drawing..." << endl;
    }
};

```

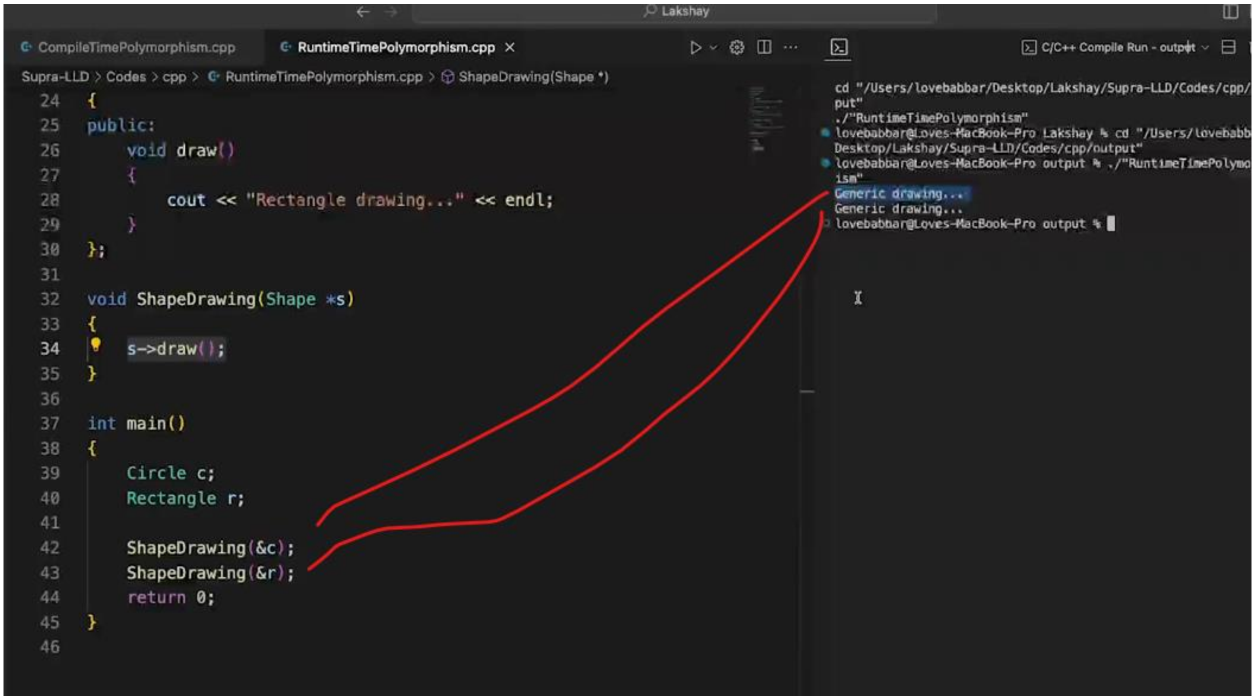


```cpp
class Rectangle : public Shape
{
public:
    void draw()
    {
        cout << "Rectangle drawing..." << endl;
    }
};

void ShapeDrawing(Shape *s)
{
    s->draw();
}

```


Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > main()

```cpp
        {
            cout << "Rectangle drawing..." << endl;
        }
};

void ShapeDrawing(Circle *s)
{
    s->draw();
}

int main()
{
    Circle c;
    Rectangle r;
    ShapeDrawing(&c);
    return 0;
}
```

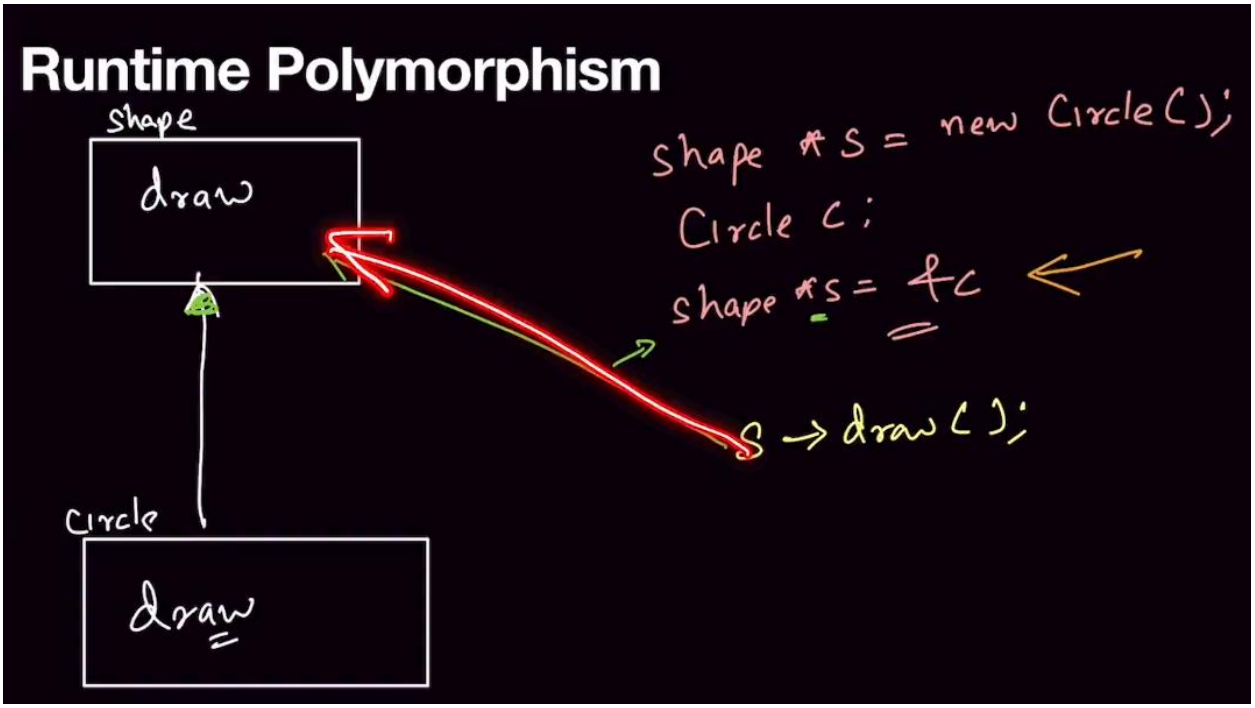
```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/
put"
./"RuntimeTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/User
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"Runtime
ism"
Circle drawing...
lovebabbar@Loves-MacBook-Pro output %
```


Lakshay
CompileTimePolymorphism.cpp
RuntimeTimePolymorphism.cpp
C/C++ Compile Run - output
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > ShapeDrawing(Shape *)

```cpp
{
public:
    void draw()
    {
        cout << "Rectangle drawing..." << endl;
    }
};

void ShapeDrawing(Shape *s)
{
    s->draw();
}

int main()
{
    Circle c;
    Rectangle r;

    ShapeDrawing(&c);
    ShapeDrawing(&r);
    return 0;
}
```

```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/
put"
./"RuntimeTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lovebabb
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeTimePolymo
ism"
Generic drawing...
Generic drawing...
lovebabbar@Loves-MacBook-Pro output %
```

Iss code mai problem hai kyuki hum ne child ka address send kiya hai shape class ke pointer ko.



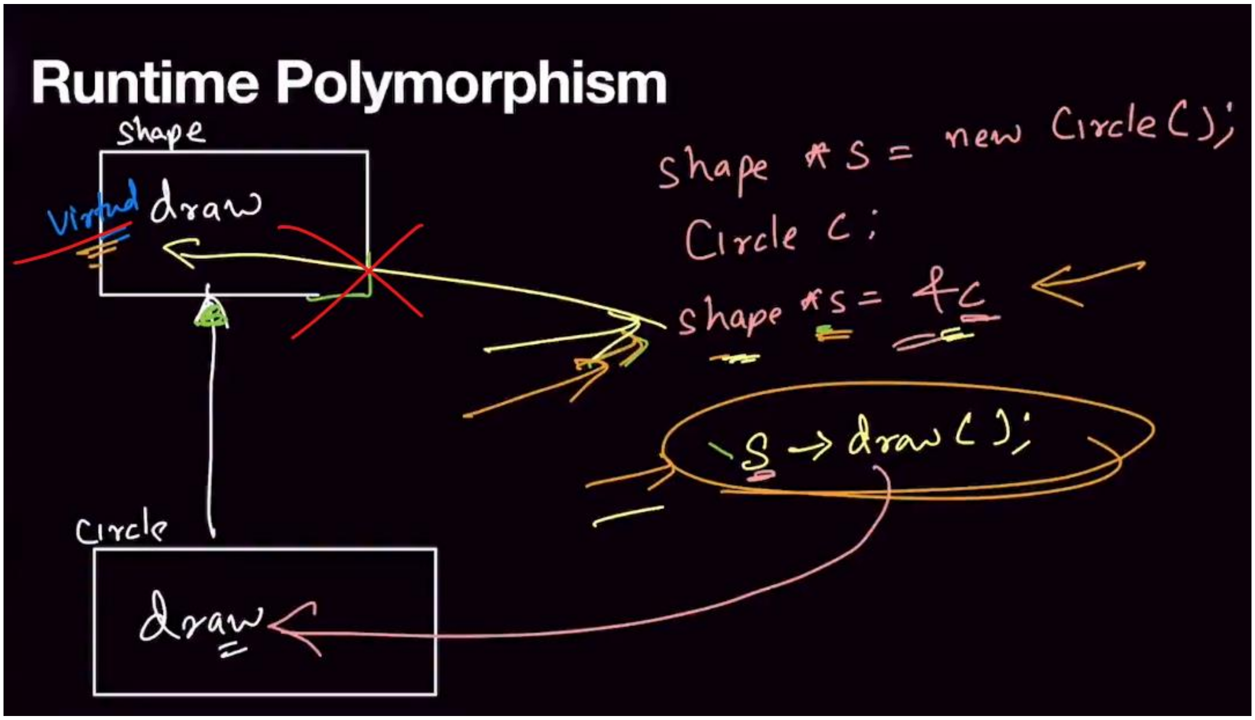
Runtime Polymorphism

```
Shape *s = new Circle();
Circle c;
shape *s = &c
s -> draw();
```

Iss mai shape ka draw()

Compliler ne dekha ki shape class ka object bana hai toh shape class ke draw() ko call kar dega iss ko Early binding kahte hai (compile time pe binding ho gayi)

Circle c;
Shape *s=&c;
upcasting : parent class ka pointer child class ko hold kar raha hai

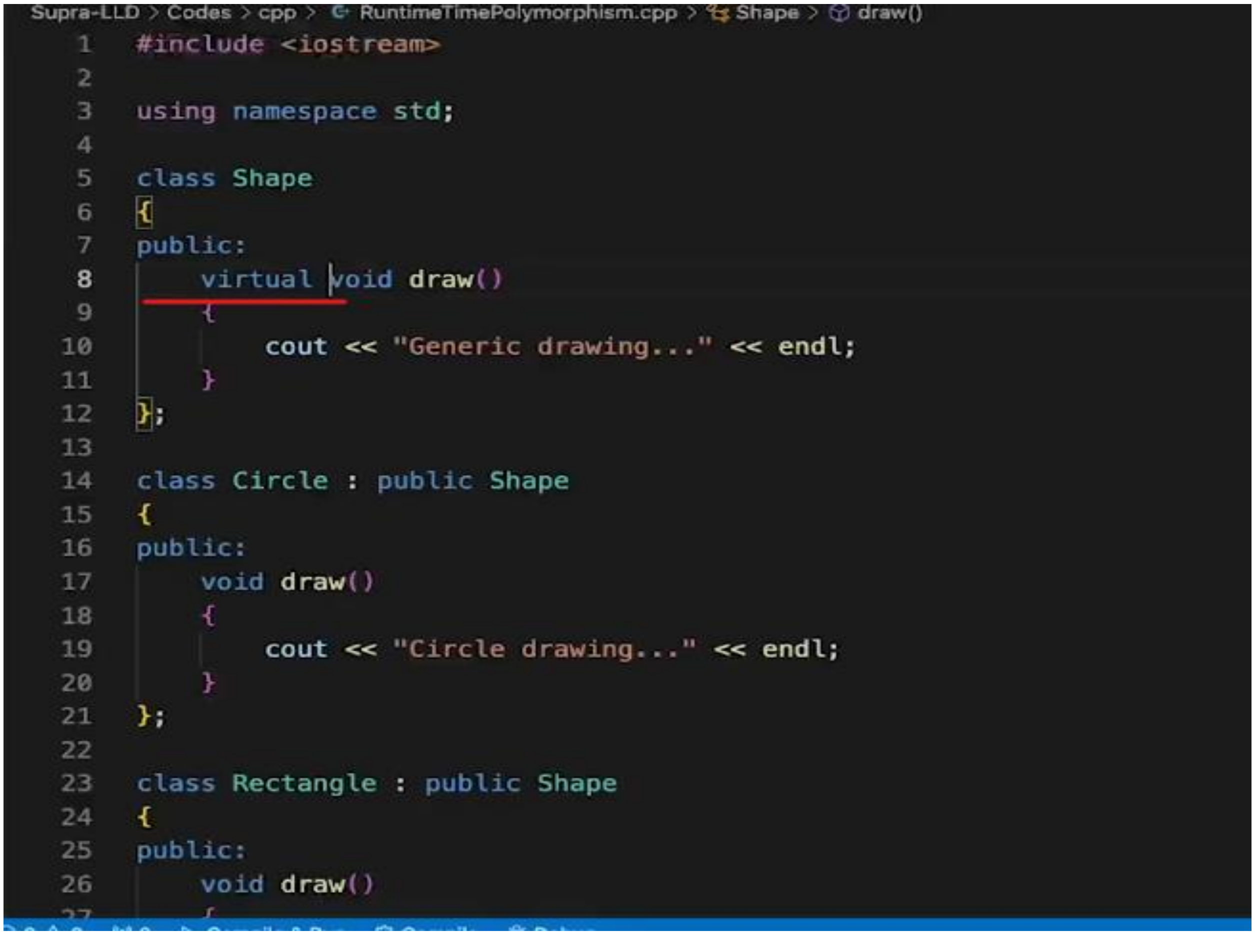


Runtime Polymorphism

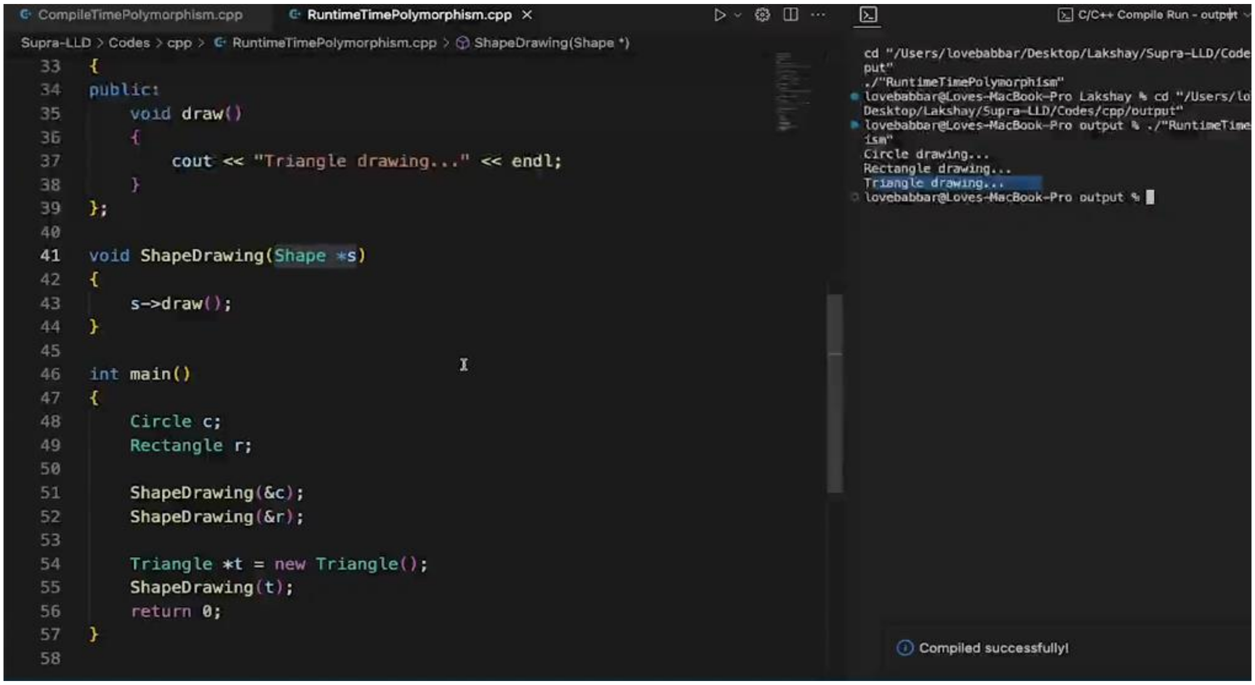
Runtime pe decision hoga ki kis menod ko call karna ? shape class ke method virtual ho gaya hai aur shape class ka object child class ke address ko hold kiya ,to hiss mai child class ka mehod call hoga aur isko hum LATE binding ya runtime polymorphism bhi kahte hai.

Late binding means runtime pe binding hogi.

Early binding means compile time pe binding hogi.


Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > Shape > draw()

```cpp
#include <iostream>

using namespace std;

class Shape
{
public:
    virtual void draw()
    {
        cout << "Generic drawing..." << endl;
    }
};

class Circle : public Shape
{
public:
    void draw()
    {
        cout << "Circle drawing..." << endl;
    }
};

class Rectangle : public Shape
{
public:
    void draw()
    {
```


CompileTimePolymorphism.cpp
RuntimeTimePolymorphism.cpp
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > ShapeDrawing(Shape *)
C/C++ Compile Run - output

```cpp
{
public:
    void draw()
    {
        cout << "Triangle drawing..." << endl;
    }
};

void ShapeDrawing(Shape *s)
{
    s->draw();
}

int main()
{
    Circle c;
    Rectangle r;

    ShapeDrawing(&c);
    ShapeDrawing(&r);

    Triangle *t = new Triangle();
    ShapeDrawing(t);
    return 0;
}

```

```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Code
put"
./"RuntimeTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lo
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeTime
ism"
Circle drawing...
Rectangle drawing...
Triangle drawing...
lovebabbar@Loves-MacBook-Pro output %
```

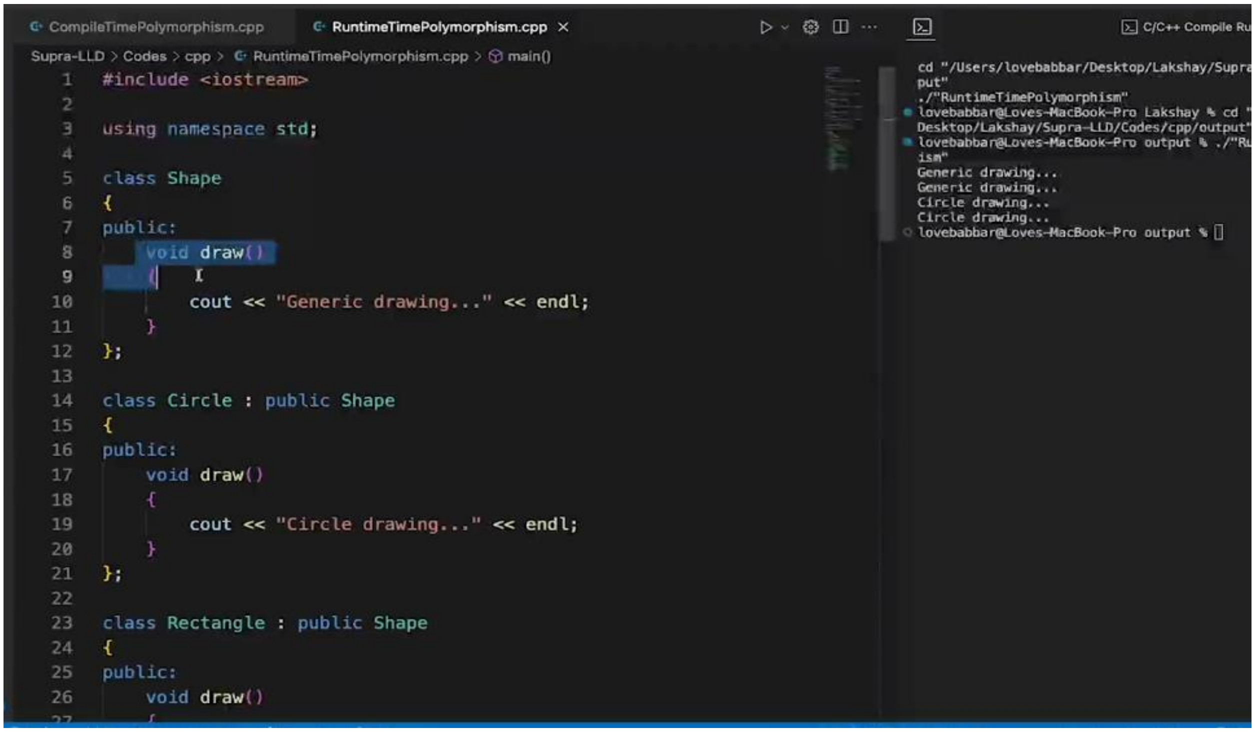
Compiled successfully!

Without virtual method bana hai shape class ke method mai

Shape *s =new shape ()

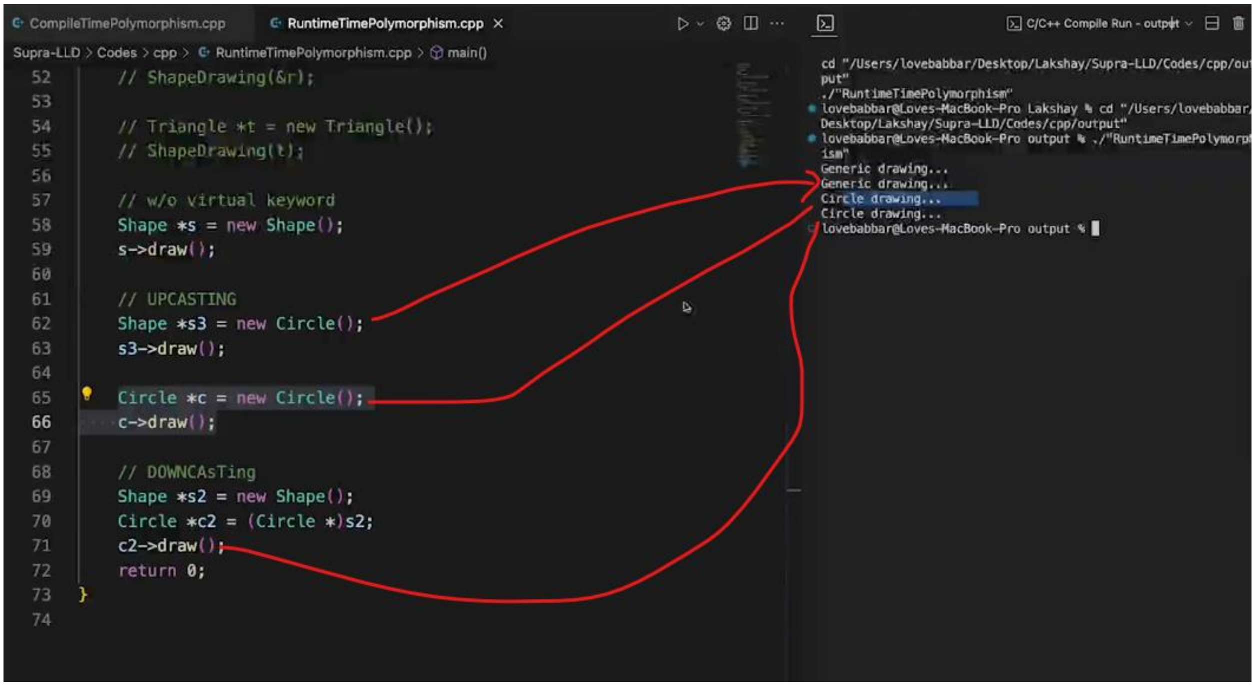
s.draw()

shape class ka draw() call hoga.


CompileTimePolymorphism.cpp
RuntimeTimePolymorphism.cpp
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > main()
C/C++ Compile Ru

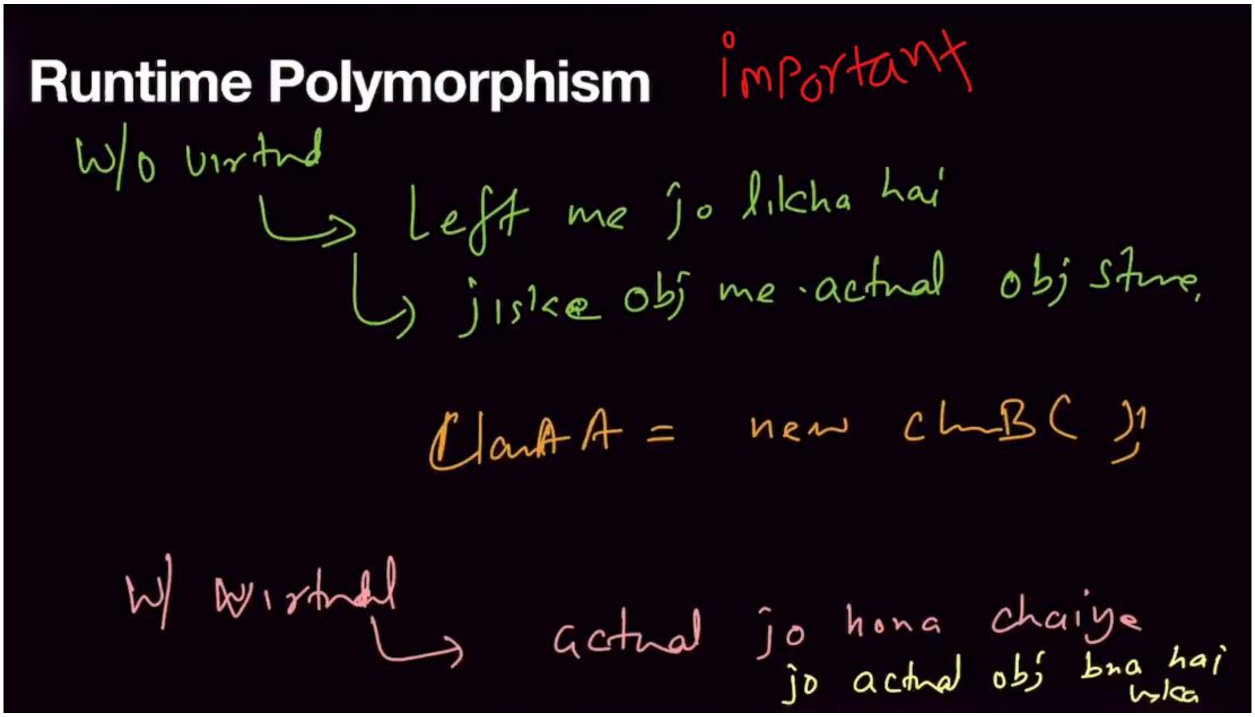
```cpp
#include <iostream>

using namespace std;

class Shape
{
public:
    void draw()
    {
        cout << "Generic drawing..." << endl;
    }
};

class Circle : public Shape
{
public:
    void draw()
    {
        cout << "Circle drawing..." << endl;
    }
};

class Rectangle : public Shape
{
public:
    void draw()
    {
```

```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra
put"
./"RuntimeTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"Ru
ism"
Generic drawing...
Generic drawing...
Circle drawing...
Circle drawing...
lovebabbar@Loves-MacBook-Pro output % 
```


CompileTimePolymorphism.cpp
RuntimeTimePolymorphism.cpp
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > main()
C/C++ Compile Run - output

```cpp
    // ShapeDrawing(&r);

    // Triangle *t = new Triangle();
    // ShapeDrawing(t);

    // w/o virtual keyword
    Shape *s = new Shape();
    s->draw();

    // UPCASTING
    Shape *s3 = new Circle();
    s3->draw();

    Circle *c = new Circle();
    c->draw();

    // DOWNCAsTing
    Shape *s2 = new Shape();
    Circle *c2 = (Circle *)s2;
    c2->draw();
    return 0;
}
```

```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/out
put"
./"RuntimeTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Users/lovebabbar/
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"RuntimeTimePolymorp
ism"
Generic drawing...
Generic drawing...
Circle drawing...
Circle drawing...
lovebabbar@Loves-MacBook-Pro output % 
```



Runtime Polymorphism

important

W/o virtual

↳ Left me jo likha hai

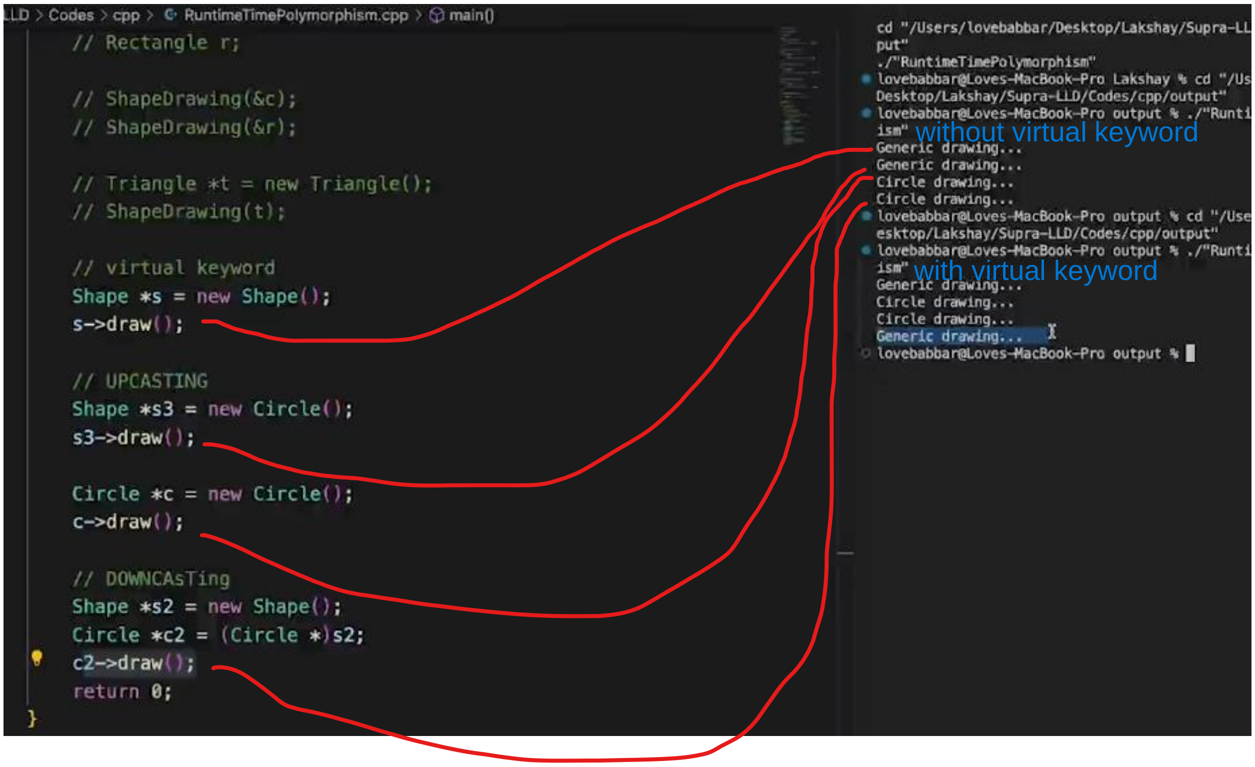
↳ jiske obj me actual obj store,

ClassA A = new clsB();

W/ virtual

↳ actual jo hona chaiye jo actual obj bna hai uska

With virual keyword

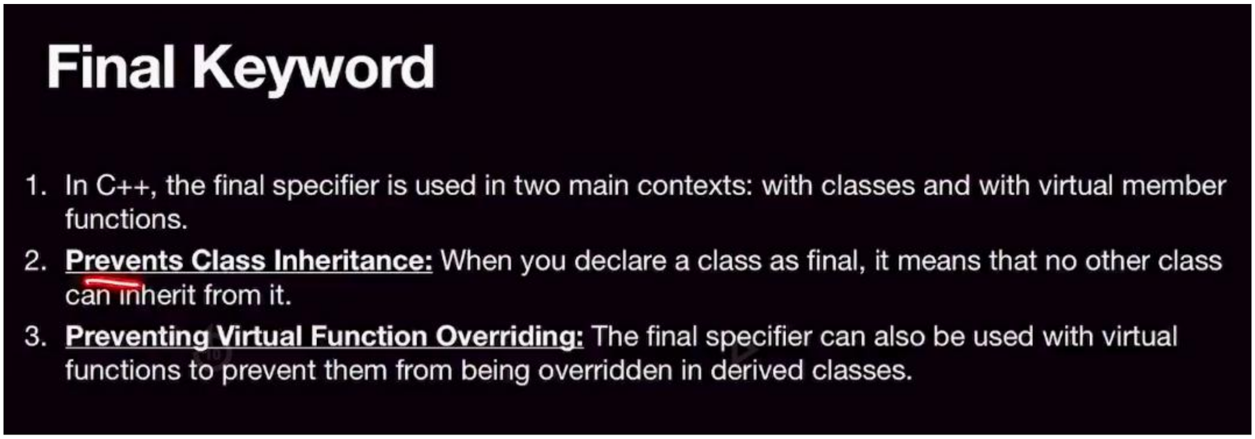


```cpp
    // Rectangle r;

    // ShapeDrawing(&c);
    // ShapeDrawing(&r);

    // Triangle *t = new Triangle();
    // ShapeDrawing(t);

    // virtual keyword
    Shape *s = new Shape();
    s->draw();

    // UPCASTING
    Shape *s3 = new Circle();
    s3->draw();

    Circle *c = new Circle();
    c->draw();

    // DOWNCASTing
    Shape *s2 = new Shape();
    Circle *c2 = (Circle *)s2;
    c2->draw();
    return 0;
}
```

```
cd "/Users/lovebabbar/Desktop/Lakshay/Supra-LL
put"
./"RuntimeTimePolymorphism"
lovebabbar@Loves-MacBook-Pro Lakshay % cd "/Us
Desktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"Runti
ism" without virtual keyword
Generic drawing...
Generic drawing...
Circle drawing...
Circle drawing...
lovebabbar@Loves-MacBook-Pro output % cd "/Use
esktop/Lakshay/Supra-LLD/Codes/cpp/output"
lovebabbar@Loves-MacBook-Pro output % ./"Runti
ism" with virtual keyword
Generic drawing...
Circle drawing...
Circle drawing...
Generic drawing...
lovebabbar@Loves-MacBook-Pro output %
```

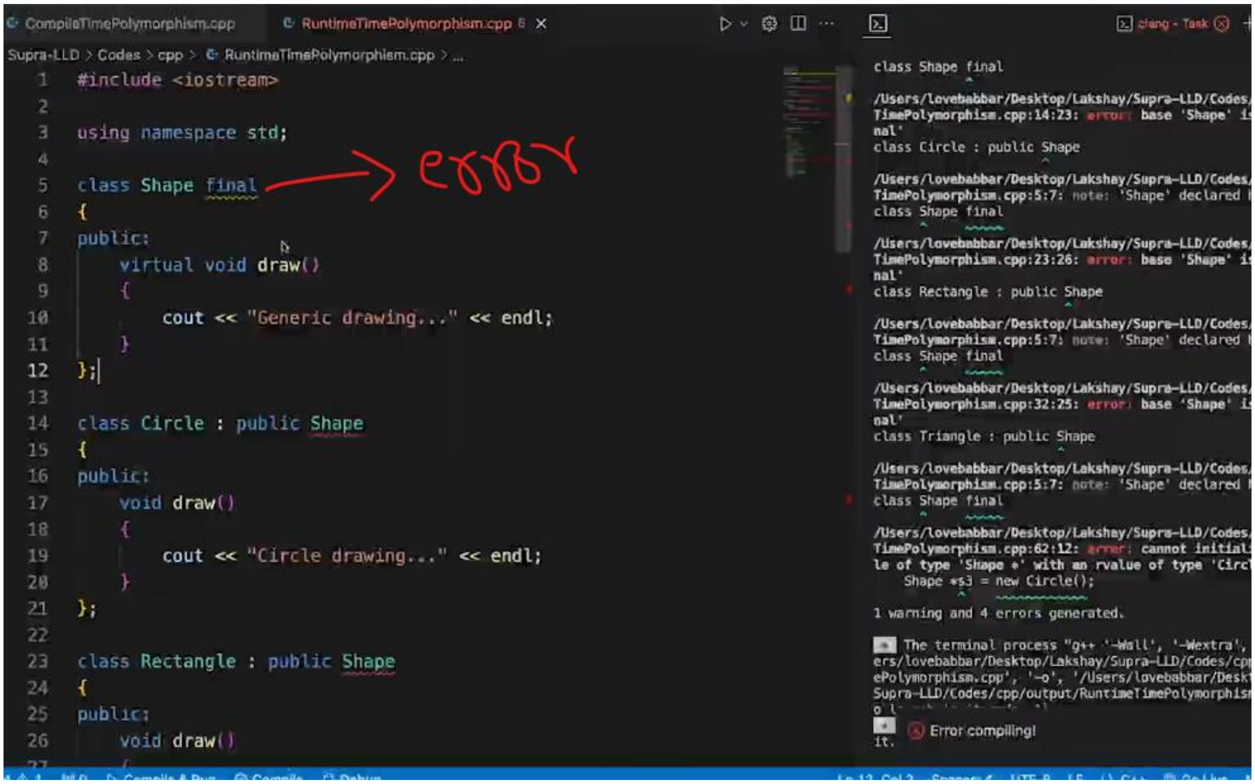
---

shape* s=new shape();
Circle * C2=(Circle *)s;
downcasting : child ka pointer parent ke object ko hold kiya hai;



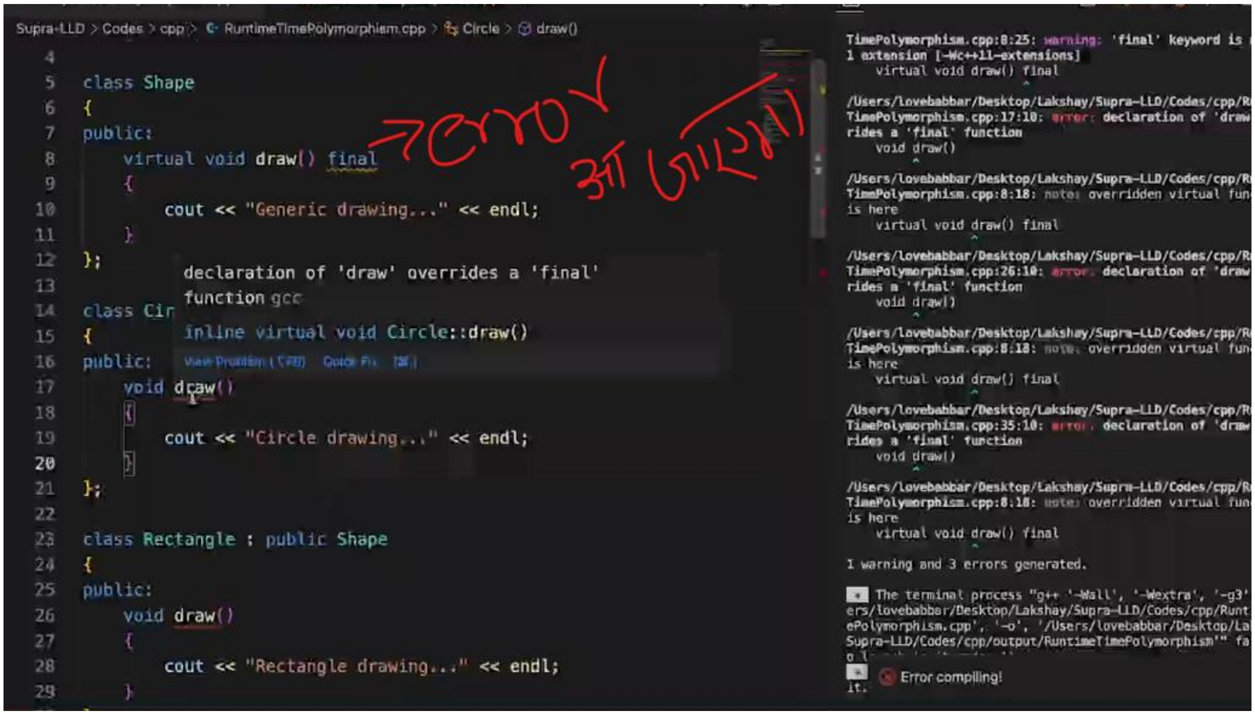
Final Keyword

1. In C++, the final specifier is used in two main contexts: with classes and with virtual member functions.
2. **Prevents Class Inheritance:** When you declare a class as final, it means that no other class can inherit from it.
3. **Preventing Virtual Function Overriding:** The final specifier can also be used with virtual functions to prevent them from being overridden in derived classes.


CompileTimePolymorphism.cpp
RuntimeTimePolymorphism.cpp 6
Supra-LLD > Codes > cpp > RuntimeTimePolymorphism.cpp > ...
error
clang - Task

```cpp
#include <iostream>

using namespace std;

class Shape final
{
public:
    virtual void draw()
    {
        cout << "Generic drawing..." << endl;
    }
};

class Circle : public Shape
{
public:
    void draw()
    {
        cout << "Circle drawing..." << endl;
    }
};

class Rectangle : public Shape
{
public:
    void draw()
```

```
class Shape final
            ^
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/
TimePolymorphism.cpp:14:23: error: base 'Shape' is
nal'
class Circle : public Shape
                      ^
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/
TimePolymorphism.cpp:5:7: note: 'Shape' declared h
class Shape final
      ^     ~~~~~
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/
TimePolymorphism.cpp:23:26: error: base 'Shape' is
nal'
class Rectangle : public Shape
                         ^
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/
TimePolymorphism.cpp:5:7: note: 'Shape' declared h
class Shape final
      ^     ~~~~~
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/
TimePolymorphism.cpp:32:25: error: base 'Shape' is
nal'
class Triangle : public Shape
                        ^
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/
TimePolymorphism.cpp:5:7: note: 'Shape' declared h
class Shape final
      ^     ~~~~~
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/
TimePolymorphism.cpp:62:12: error: cannot initiali
le of type 'Shape *' with an rvalue of type 'Circ
    Shape *s3 = new Circle();
           ^    ~~~~~~~~~~~~
1 warning and 4 errors generated.

 *  The terminal process "g++ '-Wall', '-Wextra', 
ers/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp
ePolymorphism.cpp', '-o', '/Users/lovebabbar/Deskt
Supra-LLD/Codes/cpp/output/RuntimeTimePolymorphism
o l
 *  Error compiling!
it.
```



Supra-LLD > Codes > cpp > RuntimeTimePolymorphiem.cpp > Circle > draw()

```cpp

class Shape
{
public:
    virtual void draw() final
    {
        cout << "Generic drawing..." << endl;
    }
};

class Cir
{
public:
    void draw()
    {
        cout << "Circle drawing..." << endl;
    }
};

class Rectangle : public Shape
{
public:
    void draw()
    {
        cout << "Rectangle drawing..." << endl;
    }
```

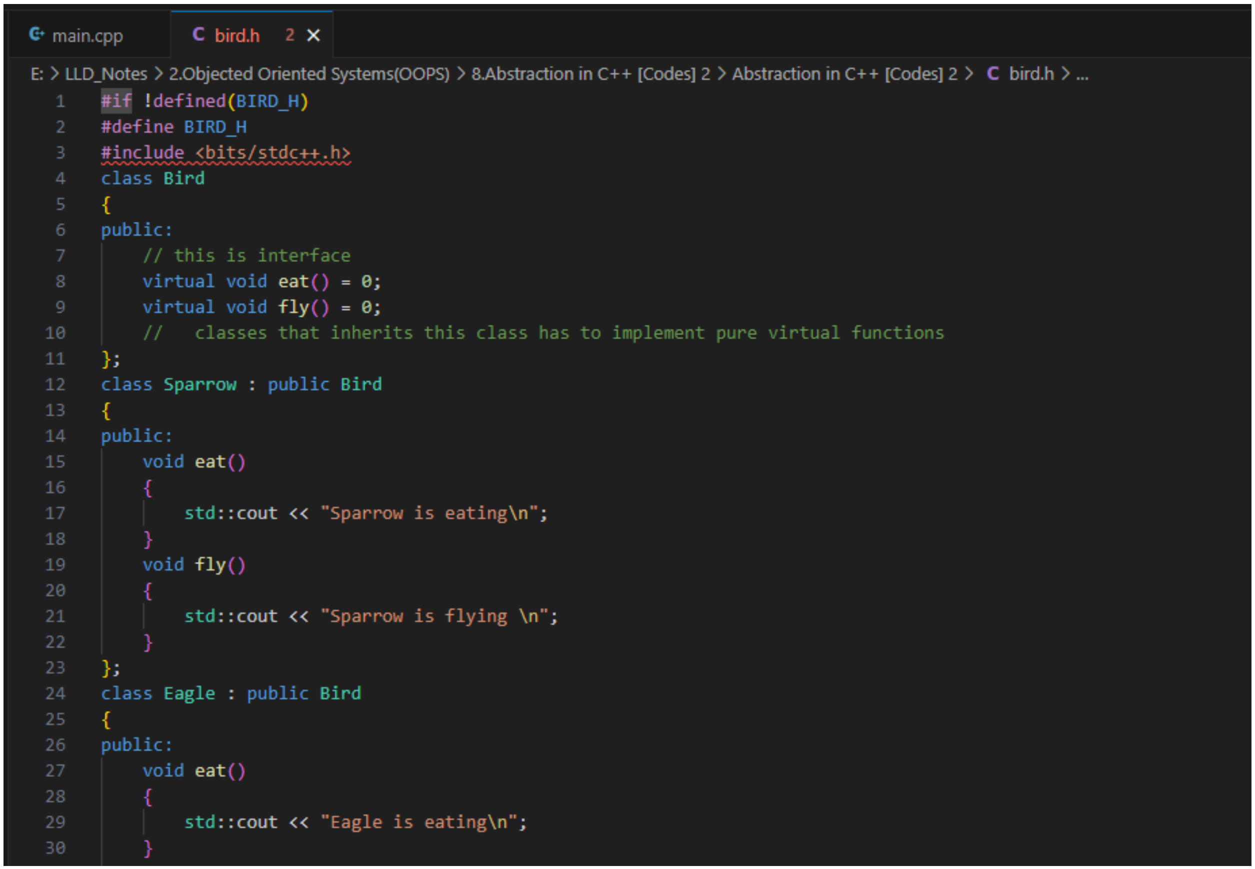
→error आ जायगा

```
declaration of 'draw' overrides a 'final'
function gcc
inline virtual void Circle::draw()
```

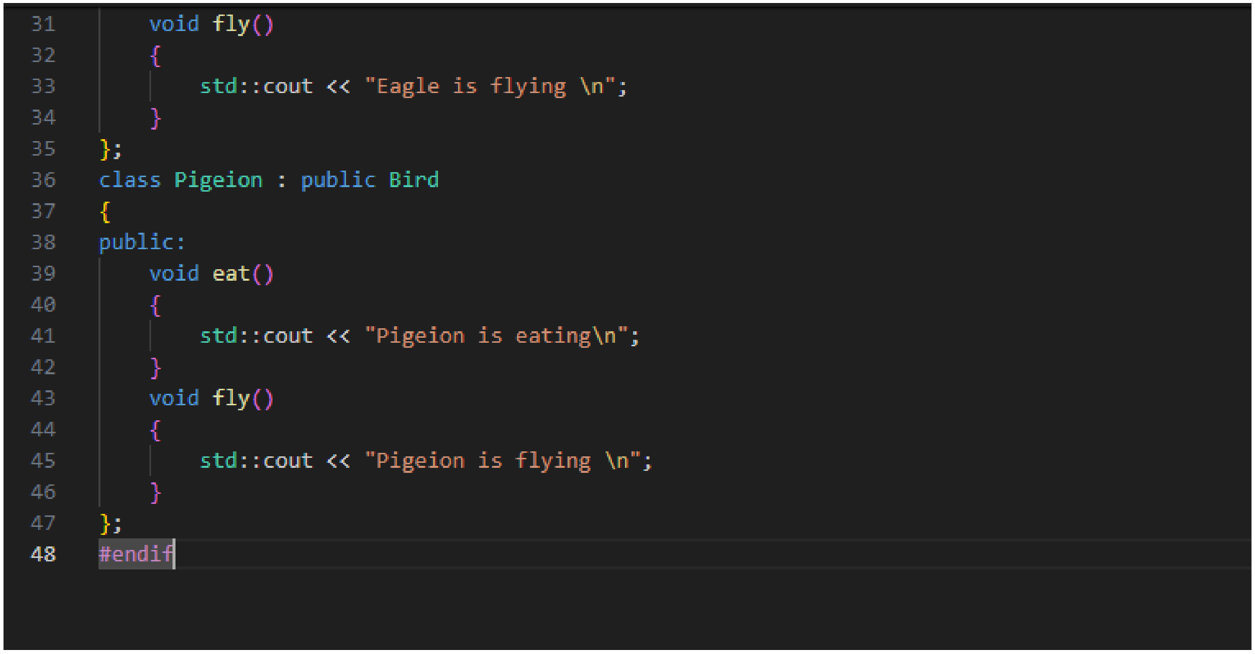
```
TimePolymorphism.cpp:8:25: warning: 'final' keyword is
1 extension [-Wc++11-extensions]
    virtual void draw() final
                        ^
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/R
TimePolymorphism.cpp:17:10: error: declaration of 'draw
rides a 'final' function
    void draw()
         ^
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/R
TimePolymorphism.cpp:8:18: note: overridden virtual fun
is here
    virtual void draw() final
                 ^
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/R
TimePolymorphism.cpp:26:10: error: declaration of 'draw
rides a 'final' function
    void draw()
         ^
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/R
TimePolymorphism.cpp:8:18: note: overridden virtual fun
is here
    virtual void draw() final
                 ^
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/R
TimePolymorphism.cpp:35:10: error: declaration of 'draw
rides a 'final' function
    void draw()
         ^
/Users/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/R
TimePolymorphism.cpp:8:18: note: overridden virtual fun
is here
    virtual void draw() final
                 ^
1 warning and 3 errors generated.

 *  The terminal process "g++ '-Wall', '-Wextra', '-g3'
ers/lovebabbar/Desktop/Lakshay/Supra-LLD/Codes/cpp/Runt
ePolymorphism.cpp', '-o', '/Users/lovebabbar/Desktop/La
Supra-LLD/Codes/cpp/output/RuntimeTimePolymorphism'" fa
 *  Error compiling!
it.
```

---

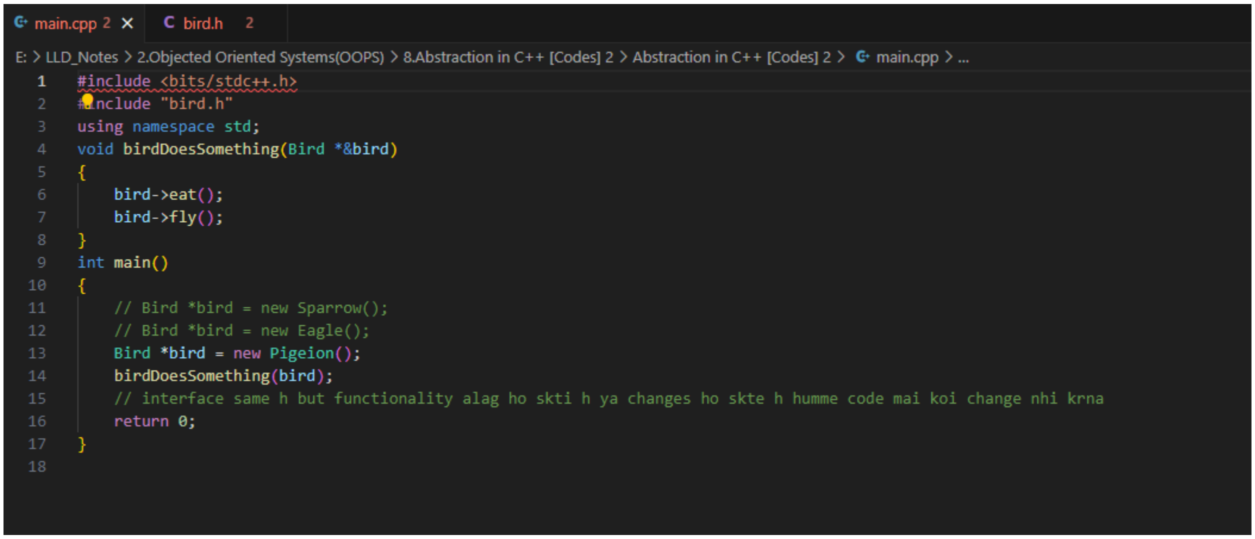
8. Abstraction In C++


main.cpp
bird.h 2
E: > LLD_Notes > 2.Objected Oriented Systems(OOPS) > 8.Abstraction in C++ [Codes] 2 > Abstraction in C++ [Codes] 2 > bird.h > ...

```cpp
#if !defined(BIRD_H)
#define BIRD_H
#include <bits/stdc++.h>
class Bird
{
public:
    // this is interface
    virtual void eat() = 0;
    virtual void fly() = 0;
    //   classes that inherits this class has to implement pure virtual functions
};
class Sparrow : public Bird
{
public:
    void eat()
    {
        std::cout << "Sparrow is eating\n";
    }
    void fly()
    {
        std::cout << "Sparrow is flying \n";
    }
};
class Eagle : public Bird
{
public:
    void eat()
    {
        std::cout << "Eagle is eating\n";
    }
```



```cpp
    void fly()
    {
        std::cout << "Eagle is flying \n";
    }
};
class Pigeion : public Bird
{
public:
    void eat()
    {
        std::cout << "Pigeion is eating\n";
    }
    void fly()
    {
        std::cout << "Pigeion is flying \n";
    }
};
#endif
```


main.cpp 2
bird.h 2
E: > LLD_Notes > 2.Objected Oriented Systems(OOPS) > 8.Abstraction in C++ [Codes] 2 > Abstraction in C++ [Codes] 2 > main.cpp > ...

```cpp
#include <bits/stdc++.h>
#include "bird.h"
using namespace std;
void birdDoesSomething(Bird *&bird)
{
    bird->eat();
    bird->fly();
}
int main()
{
    // Bird *bird = new Sparrow();
    // Bird *bird = new Eagle();
    Bird *bird = new Pigeion();
    birdDoesSomething(bird);
    // interface same h but functionality alag ho skti h ya changes ho skte h humme code mai koi change nhi krna
    return 0;
}
```

Abstraction ya Inteface class ka OBJECT nahi banata hai .
